Content Based Graph Visualization of Audio Data for Music Library Navigation

Chris Muelder, Thomas Provan, and Kwan-Liu Ma
Computer Science Dept.
University of California Davis
Davis, CA, USA
muelder|ma@cs.ucdavis.edu,tcprovan@ucdavis.edu

***Abstract*—As a user's digital music collection grows, it can become difficult to navigate. Music library programs aid in this task by organizing music according to tags such as artist or title. However these generally utilize a text based interface, and they do not take into account the content of the music itself. As such, they do not handle untagged or mistagged music well. Automated metrics exist, but are not as widely used since they have the potential to be unreliable. This paper presents a graph-based visual interface for exploring a library of music based on analysis of the content of the music rather than tag information, which allows the user to navigate a music library thematically.**

***Keywords*-Information Visualization; Graph Visualization; Music;**

I. Introduction

Compressed digital media formats have been steadily becoming more popular due to their small size and convenience, and data storage sizes have been increasing steadily. The combination of these factors has led to an explosion of audio data availability. The rise of compressed digital music (such as the ubiquitous mp3 format) has enabled users to store hundreds or even thousands of songs on their computer or portable device. As audio collections grow to this size or beyond, organization and navigation can become difficult.

Music library software, such as iTunes [1], provide an automated way to index and organize music according to tagged information in the music files such as artist or album names. The user can then search and navigate this indexed library to create playlists. However, this requires that the entire library is properly tagged. In the event that music files get tagged incorrectly or simply lack certain fields, they can get lost in the library. This can be quite common, particularly if the user is entering the information by hand. Automated, crowd-sourced tag entry and tag completion or correction can help if the audio files are all popular, but obscure music such as that of local artists or other, non-music audio sources are often not in these tag databases.

Most music library programs are text based, requiring the user to read and mentally process the tag information for each song. This can make it tedious to go through large numbers of songs to make a playlist of songs that work well together. That is, if the user wants to make a playlist that fits a particular mood, either they must either remember every song in the collection and jumps straight to the ones that fit the mood (which defeats the purpose of the music library software), or they must go through the entire music collection (or at least a large portion of it) and read through the songs to find ones that fit their mood.

In many cases, the audio data can completely lack useful tag information. Security and intelligence analysts can encounter large collections of raw audio to sift through. For instance, identification of unknown participants in a conversation would require analyzing the raw audio logs. Collections of these audio logs could be too large to effectively listen through one at a time, and it can be difficult to recollect and match up similar sounding voices. Speech recognition can be useful for converting audio data into transcripts that can be searched through, but this does not help for identifying the voices themselves.

This paper presents a visual approach to audio collection organization and navigation which is based entirely on the content of the audio. This approach is based on a combination of Fourier analysis techniques common in speech recognition algorithms with a graph paradigm to present the relationships between different songs to the user.

A. Related Work

Many existing music tools are essentially text based. The industry standard is currently music library based applications such as iTunes [1], which index music according to tag information such as artist, album, genre, and more. While this is much better than manually indexing large numbers of music files by hand, it is still completely reliant on accurate tagging information, and its text based interface can be tedious when the number of songs gets large. iTunes has recently added a 'Genius Mixes' feature, which collects music information, then sends it to a central server which analyses the song information to create playlists and recommend songs outside the library. While this can be useful, it is currently a black-box process, which makes it difficult to evaluate. Also, Apple's collection of user's information raises questions of privacy and anonymity, which makes some users wary. Similar to the 'Genius Mixes' is the Pandora internet radio approach, which takes an initial song and some very simple user input to select music according to a rather extensive number of centralized tags created by

experts. While this can be a very effective way to expose users to music they might not have encountered before, it does not aid a user in navigating the music they have already added to their collection. It is also tag based, and maintaining such a large tag database is a large task in itself. CUIDADO [33] is another recommendation based system which predates Pandora. It uses derived information and not just tagged data, but it is still text based. MusicLens [18] is a text based interface used to filter a large collection of music according to user selections.

There are some graph-based visualizations for music exploration, but these mostly focus on browsing by artist. LivePlasma [13] and TuneGlue [30] use graphs to present the relationships between artists based on statistical information such as Amazon.com's users' purchase histories. Musicovery [20] is an extension of LivePlasma which includes streaming music. While it introduces a 'mood' interface which filters songs according to tempo and tone, it too is based primarily on sales statistics. Barcelona Music & Audio Technologies (BMAT) [3] presents a graph of artists very similar to the others, which is based on sales and blog statistics as well as content derived information.

Several music library visualizations approach the problem with a geographic metaphor. Islands of Music [23], [26], nepTune [10], PlaySOM [21] and Globe of Music [29] all use self organizing maps to arrange songs with similar audio characteristics. This is then presented to the user through a heat map or, in the case of Globe of Music, pictures on a globe.

Still other approaches present data in more abstract spaces. Gnod's music-map [19] uses input from previous users to define a database of related artists, which it then uses to present artists similar to the artists the current user inputs. Donaldson's MyStrands [6] uses songs' playlist co-occurrence to match up artists or songs. Music Rainbow [24] and MusicSun [25] use a combination of audio feature comparisons with web-crawling and word-based comparisons to sort artists in order to present an intuitive visualization. MusicSim [5] presents the user with clusters of music, which the user can manually refine.

The most closely relevant related works are probably Lamere's "Search Inside the Music" [11] and Lillie's MusicBox [12]. "Search inside the music" uses audio analysis to compare music files and plot them in 3d space according to a dimensionality reduction technique. While this is completely based on properties of the music itself regardless of tag information, the results are limited by the layout which is very dependent on the classification method. In particular, the authors note that their approach does not work well when the classification methods produce very tight clusters. Similarly, MusicBox uses a dimension reduction technique to generate a layout. However, MusicBox also incorporates tag based information. In the absence of accurate tag data, it falls back on content based comparisons, but it is still constrained by the choice of layout. As Hoashi et al. have shown [8], such dimensionality reduction based layouts can be effective, but unintuitive and unfamiliar to many users. Our work proposes a more intuitive graph-based view which produces results similar to these previous works by analyzing music information at a very low level without depending on tag information and uses this analysis to generate a visualization which avoids overlap issues generated by dimensional reduction based layouts.

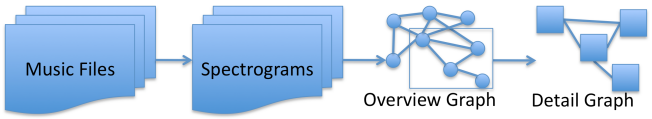


Figure 1. The overall process. Music files are transformed into spectrogram-based signatures, then compared to form a relationship graph. The user can then select subgraphs to view in more detail.

When analyzing the contents of individual audio files, common techniques involve Fourier analysis. Thus, many of the aforementioned works utilize discrete Fourier transforms (DFT) to convert the sampled audio waveform to frequency space. From the frequency information, Mel-frequency cepstrums [2], [4], [14], [27], [28], [31] are another common technique for deriving signatures of audio data for comparison. We utilize these algorithms to analyze and compare music files against each other. We then employ and extend the visualization techniques of [17] in order to create an intuitive representation and interface. And we add some of the interaction techniques of [32] to support query based interactions and fisheye lens techniques [7] for detail exploration.

II. Methodology

Our approach consists of a graph-based representation of similarities derived from frequency analysis. Unlike in our previous work [17], this approach does not use wavelets. Rather, it uses discrete Fourier transforms to convert the music files to frequency space in the form of spectrograms. These spectrograms are then statistically reduced to signatures and compared against each other via a similarity metric. This generates a matrix of similarity values which we represent with a graph. The user can then interact with this graph by selecting subgraphs to populate a second detail view, which can then be refined, adjusted, and/or exported as a playlist. The overall process is depicted in Figure 1.

A. Fourier Analysis and MFCCs

The audio waveforms in music cannot be directly compared, as issues such as compression artifacts, noise, and phase shifts prevent direct correlation. So the first step in analyzing the content of the music data is to transform it to a space in which it can be compared. The wavelet scalogram method of [17] was tried, but it did not produce effective results, so another method was needed. One of the

most common methods for analyzing audio data is a Fourier transform, which converts a data series to frequency space. However, converting the entire waveform to frequency space can lose large scale features and would require $O(n\ log\ n)$ computation where n is the total number of samples in the song. Fourier based spectrograms offer an improvement on these points. Spectrograms work by performing Fourier transforms on small subsections of the song at a time. The result of this is that the spectrogram produces a two dimensional representation of the data which plots frequency against time. This frequency information can subsequently be used to derive a 'fingerprint' of the signal that is relatively invariant to changes due to phase or noise. Also, spectrograms require only $O(n\ log\ w)$ calculation, where n is the length of the song and w is the length of the window. We utilize spectrograms as a first step in analyzing the audio. In order to generate the spectrogram for each song, we first divide the song into discrete sequential chunks of a fixed window size. Then, a FFT is applied to each chunk to transform it to frequency space.

One result of using spectrograms is that frequencies high enough to fit within the window size are transformed to frequency space, while frequencies which are lower than the frequency defined by the window size remain in the time domain. In audio data, this is an intuitive representation, as the human ear only interprets frequencies which are above a certain threshold (around 20Hz) as pitch. Any frequencies below that correspond to more temporal aspects such as rhythm or the tempo of a song. Now as the window size w is flexible algorithmically, we can choose one which corresponds to this natural threshold. As most audio is sampled at 44KHz, this window size is around 2,200 samples. For computational simplicity, we adjusted this to the nearest power of 2 - specifically we set the window size to $2,048$ samples, which captures frequencies of $21.48Hz$ and above, while leaving lower frequency patterns in the time domain.

While spectrograms transform the audio signal into a more intuitive form, they are still too large and noisy for direct comparison. We found experimentally that simple statistical metrics such as finding the averages of or the standard deviations of amplitudes at each frequency produces fairly effective signatures, but these are still quite large (about 1000 dimensions). Also, the linear scale in frequency space can potentially over-emphasize the contributions of high frequencies, as audio is perceived with the frequencies on a logarithmic scale. One commonly used solution to this problem is to transform the data to the mel scale, which is a frequency mapping that is perceived as linear to the ear. From there, the a common technique is to take the mel-frequency cepstrum (MFC) of the audio signal [2], [4], [14], [27], [28]. This approach is used by many existing audio fingerprinting libraries [27], including the freely available, open-source library Marsyas [31].

The MFC of a signal is a set of mel-frequency cepstral coefficients (MFCCs) which form a cepstrum (a 'spectrum of a spectrum') centered around mel frequencies which are perceptually equidistant. Specifically, it is calculated for each sample window by taking the spectrogram of the signal, resampling it on the mel scale using overlapping triangular windows, scaling the amplitudes at each mel frequency logarithmically, then taking discrete cosine transform of the resulting powers at mel frequencies as if they were a signal.

These MFCs are much smaller than the original spectrogram, as the number of mel frequencies to sample is adjustable. Most existing works and implementations seem to use 10-15 mel frequencies, so we chose to use 13 mel frequencies. However, since this calculation is done per chunk of the signal, there are 13 times the number of chunks in the data MFCC's to consider. In order to create feature vector small enough for general comparison, we aggregate the MFCCs in temporal space. Specifically, we calculate the mean and standard deviation of each MFCC over the duration of the signal as

$$Avg = \{avg_i | avg_i = \frac{\sum_{t=1}^{n} mfcc_{t,i}}{n}\}$$

$$Std = \{std_i | std_i = \sqrt{\frac{\sum_{t=1}^{n} (mfcc_{t,i} - avg_i)^2}{n}}\}$$

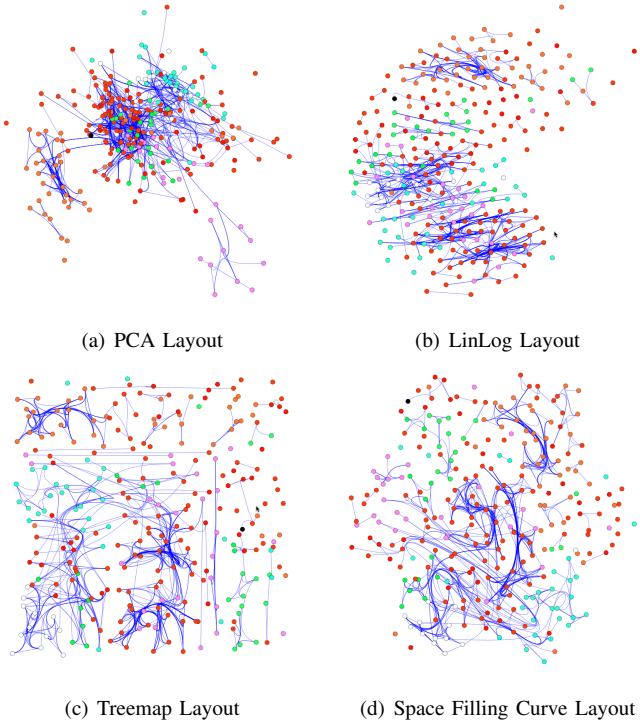
By calculating the feature vectors in this way, the average vector should capture high frequency patterns such as timbre, while the standard deviation vector should correlate with lower frequency patterns.

B. Similarity Metrics

The next step is to take the signatures and represent how well pairs of them match with a single number to be used as an edge weight. This can be done in many different ways, such as taking the arithmetic mean of the relative differences, the geometric mean of the differences, or the inverse of the Euclidean distance between the two signatures. The methods we use are calculated as follows. Let $a = (a_1, a_2, ..., a_n)$ and $b = (b_1, b_2, ..., b_n)$ be the signatures corresponding to two nodes, then the weight w of the connection is:

- Arithmetic: $w = \sum_{i=1}^{n} \frac{1 - \frac{:a_i - b_i:}{a_i + b_i}}{n}$
- Geometric: $w = \sqrt[n]{\prod_{i=1}^{n} (1 - \frac{:a_i - b_i:}{a_i + b_i})}$
- Geometric(no root): $w = \prod_{i=1}^{n} (1 - \frac{:a_i - b_i:}{a_i + b_i})$
- Inverse Euclidean: $w = \frac{1}{1 + \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}}$
- Inverse Euclidean^2: $w = \frac{1}{1 + \sum_{i=1}^{n} (a_i - b_i)^2}$

Different methods can be more or less effective with different data transformations. That is, the arithmetic and geometric means are calculated relatively, so they are more effective when there are large changes in the signatures relative to the average value. When the signatures tend to have large average values and the relative changes are small, then one of the Euclidean functions work better. In practice, we found the Euclidean metrics to work well on MFCC data.



(a) PCA Layout (b) LinLog Layout

(c) Treemap Layout (d) Space Filling Curve Layout

Figure 2. *A graph of a music library*. Different layouts produce different results, but some take more computation than others. Hierarchical edge bundling [9] is used to route the edges in order to reduce clutter.

C. Graph Representation

In order to create an overview of an entire music library, a graph paradigm was employed. In this graph, each node represents a song, and the connection between any two nodes is weighted according to the statistical similarity between them, with 0% being completely different and 100% being identical. Since this is a complete graph, a cutoff is employed to simplify the graph – any connection where the nodes match less than some user-defined threshold is dropped. This threshold ranges from 0 to 1 and is adjustable by a slider. From here, the graph is laid out on the screen in order to group similar songs together.

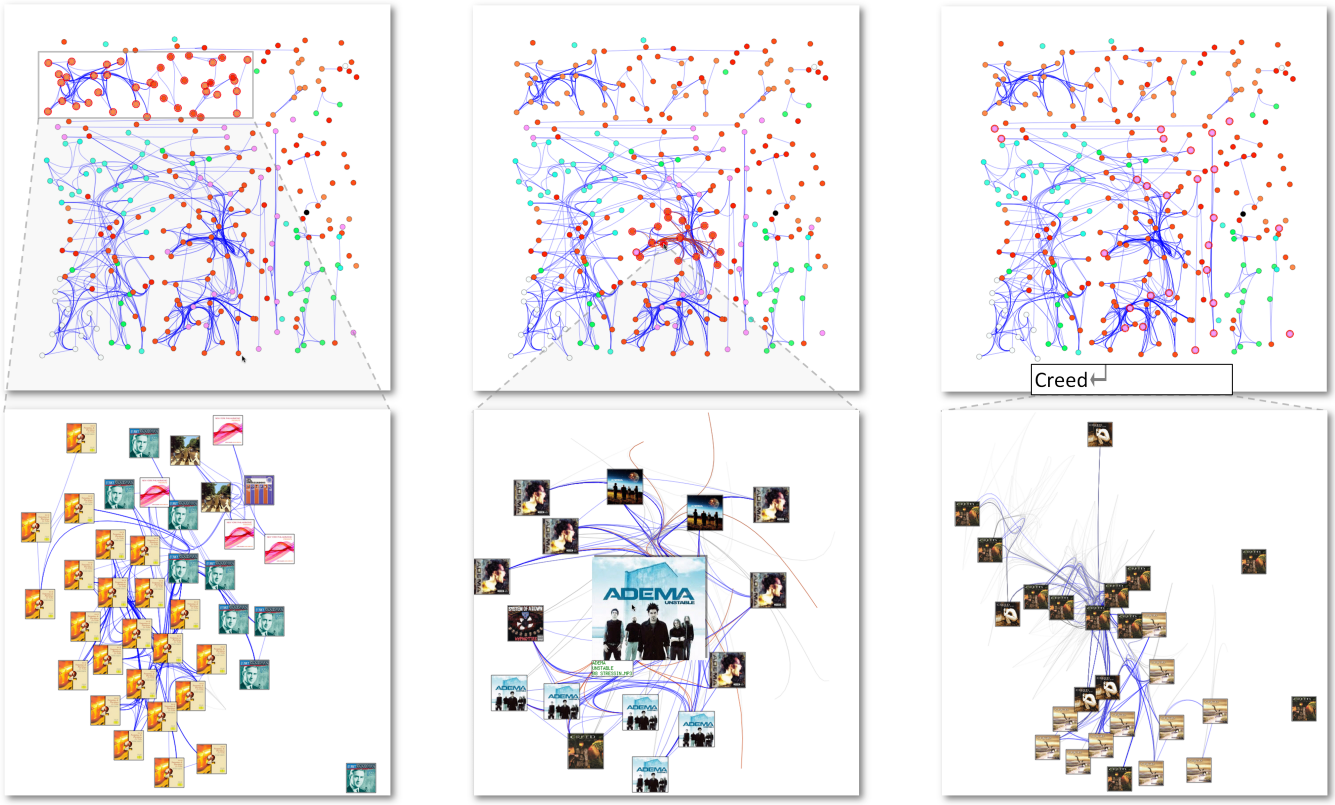
We used 4 different layout algorithms: an algebraic layout, a force directed layout, and 2 hierarchical clustering based layouts. For the algebraic layout, we used Principal Component Analysis (PCA) on the high dimensional input data to organize the data. This is quite fast, and produces results similar to existing works such as MusicBox [12]. An example is shown in Figure 2(a). However, the PCA layout runs into problems with nodes being placed very close to each other or even completely on top of each other, making them hard to discern. The force-directed layout we used was the LinLog algorithm [22]. This approach is quite slow on larger data sets, taking upwards of a minute on the larger graphs shown here, but produces the most aesthetic results. As it still works well on smaller graphs, we often use it for the detail graphs. An example is shown in Figure 2(b). Finally, we used the treemap based layout and space-filling curve layout of [16] to quickly generate efficient layouts which try to minimize clutter and collisions while still grouping clusters of songs together. The treemap based layout in particular was found to be effective for interactions such as box-selection. An example of the treemap based layout is shown in Figure 2(c) and the space-filling curve layout is shown in Figure 2(d).

In each layout, nodes with higher match percentages are placed closer to each other more than nodes that do not match as well. Thus, similar songs are clustered together. This effect can be clearly seen in Figure 2, which shows a graph of a sample library. Additional information can be shown in the color of the nodes. Any given node can be colored according to details such as artist, genre, or song length. While these do not directly affect the clustering, often after the clusters have formed, patterns can be seen in the data, such as clusters that are all the same artist or genre.

While the graph representation is relatively intuitive, the density of the network can make the graph hard to read; as the number of edges increase, the visualization becomes cluttered. One solution to this problem is the edge bundling approach of Holten [9] which routes similar edges together according to a hierarchy. As we are already generating a clustering hierarchy as part of the layout process, this same hierarchy can be used to implement edge bundling. The result of this is a much cleaner, aesthetic looking graph, while losing the ability to easily follow individual edges through the bundles. In order to alleviate this limitation, edge highlighting on node mouse-over was added to make it easier to identify neighboring nodes.

D. Detail Graph

Similar to the overview graph, the detail graph shows the relationships between different pieces of music. However, it only displays a user-selected subgraph of the data. Since the data shown in this view is generally much smaller than the entire data set, more space is available to show details. In particular, we represent each node in the graph with the album art when available, and a default image otherwise. We tried using the spectrogram of each song for the detail representation, but found it unintuitive and less useful than the album art. When the user mouses over a song in this detail graph, the detail representation is expanded to a larger size and the song's tags are presented textually in order for the user to identify it. Also, the edges connected to the node of interest are highlighted to help the user see what it connects to. As in the overview graph, there is the option to employ a fisheye lens centered on the mouse, which pushes the rest of the nodes out of the way as the user explores particular nodes of interest. Closely related nodes that were not selected are hinted at through the use of fading edges to invisible nodes, as in [32]. The user can then click on



(a) *Box Selection.* The user drags out a selection box, and its contents are used to populate the detail graph.

(b) *Individual+Neighbor Selection*. When the user selects a single node from the overview, the detail graph is populated with its immediate neighbors.

(c) *Keyword Selection.* The user can use keywords as a filter for the data. The detail graph is populated by matches.

Figure 3. *Subgraph selection and Interaction.* User-selected subgraphs are shown in a second, more detailed view, which can the be refined and explored.

these hidden nodes can be used to expand the subgraph. Alternately, if the detail graph is still too large, the user can refine it through a box selection which discards all the nodes outside of the selected box. Finally, the user can export the subgraph in whole or in part to create a standard playlist which can be opened in most media players.

The two views (overview graph and detail graph) are linked through user interaction. From the graph overview, the user can select a subset of the data to be shown in detail in the second view while highlighting the selected nodes in the original graph to show context. This can be done with a selection box, by clicking an individual node, or by entering a keyword with which the data is filtered. Examples of all three of these are shown in Figure 3. The selection box is probably the most familiar to the average user, as it is a common interaction in most desktop operating systems. The user drags out a selection box with the mouse, then a subgraph is generated from the nodes inside the selection box. Figure 3(a) shows an example of this with a fairly diverse selection. In this example, a section of the graph was selected which contains a wide variety of music, and the detail view shows how they relate to each other. For the individual node selection, the user clicks an individual node and a subgraph is generated consisting of the clicked song and all nodes whose edge to the clicked song exceed the current display threshold. Figure 3(b) shows an example of selecting one song in a dense part of the graph, and it is then focused on with the fisheye lens in the detail graph. This selection reveals that there are a large number of songs which are all quite similar to each other. However, within this dense graph, songs by the same album tend to group together, indicating that they are still more tightly connected to each other than to the rest of the graph. The third method of interaction with the overview graph is search based: when a keyword is entered, a subgraph is generated consisting of all nodes which match the keyword. This mimics the search process of traditional music library software, but still presents the results in comparison to each other. An example of this is shown in Figure 3(c), where the library was searched for “Creed”. In this example, the songs group according to album, but there is some overlap between them and some clear outliers.

III. Case Studies

We applied our technique to a music library with 30 albums by 14 artists of various genres. Subgraphs of different selections, shown in Figure 4, show some results of our approach. If we select the Beatles (Figure 4(a)) we find that the songs end up arranged mostly by album. However, there are still many connections between the albums, and even a song that is disconnected in this view. When we select the Creed albums (Figure 4(b)) there is an even clearer distinction between albums. In particular, their second album is completely separate. Interestingly, only the songs from the third album have strong ties to the rest of the library. Also of note is the song on the right side which is isolated from the rest of the network. On inspection it can be seen that this song is a lullaby, which is very different from the rest of the songs in the subgraph, even though it is tagged as being similar (same artist, album, and genre). Another interesting album is the Dr. Horrible soundtrack (Figure 4(c)). As Dr. Horrible is a musical, the songs in it are varied and involve a number of different singers and styles. While many of the songs are clustered fairly tightly, there are a few on the outside of the cluster which are not that closely tied and even pairs of songs that are completely separate. One pair of isolated songs are actually both sung by Felicia Day with piano accompaniment. The other pair consists of the 'Bad Horse' theme songs. Both pairs are quite different from the rest of the album, and this difference is distinctly visible. One other interesting example has to do with the New York Philharmonic Orchestra and its relation to jazz music (Figure 4(d)). In this example, the system mostly splits the two jazz albums apart, which makes sense as the Ken Burns album is big-band swing music, while The Ambassadors' album is more traditional jazz. What is interesting is that it also splits the Philharmonic album in half, and associates each half with one of the jazz albums. Upon inspection, it can be seen that the tracks in the half of the Philharmonic recordings associated with the big-band swing music are symphonic works, while the tracks in the half associated with the traditional jazz are actually spoken introductions to the symphonic works.

IV. Future Work

While we found the spectrogram analysis to be effective, it often became computationally time-consuming. However, this is alleviated by the fact that it is a pre-processing step. In order to fully address this problem, a stochastic sampling of the audio could be effective. One limitation in our approach is our handling of the change in MFCCs over time. Namely, by saving only the average and standard deviations of the MFCCs, we potentially lose a lot of the lower frequency patterns, such as tempo or rhythm. Applying wavelet or Fourier analysis to the MFCCs themselves over time could yield interesting new metrics which capture these properties. As our system is not dependent on the MFCC metric, it would be possible to use other audio metrics, and compare, contrast, and combine them to create more robust signatures, as in [15]. While it is very difficult to establish the correctness of approaches such as ours, it would be interesting to see how well they agree with other measures of similarity, such as tag based, or co-purchasing metrics. Alternately, a user study would be beneficial for establishing how well automated similarity metrics agree with perceptual similarity. We have not yet extended the system to handle audio collections which grow too large to fit on the screen at one time. However, the treemap based layout was developed with this kind of interaction in mind, so it should be straightforward to implement through the use of this layout. Another interesting direction would be to distribute the system on the Internet and see how other people use it. In the process, it would be very beneficial to anonymously collect the signatures of large quantities of user's music to make suggestions to the user from outside their own music library, similar to current music library systems. We also found that scalability of the overview can be an issue, so an alternate overview could be of merit. Our approach would be applicable to other forms of media such as images or video. However, complexity issues could arise due to the extra dimensions.

V. Conclusion

We have taken a graph-based approach to the analysis of large audio collections, which is founded on frequency analysis of the audio itself. The graph representation provides an intuitive visual interface for interactive navigation that does not require the user to have a predetermined goal nor accurate tag information for the audio pieces. In this paper, we present the results of applying our method to analyzing a music library of thousands of songs, but we have also employed it to a collection of speech recordings and found interesting results. It is clear that our method is extensible for the analysis of larger or more general audio data comparison tasks, which are often found in security applications. Such a visual-based method is thus appealing to analysts and can assist them to more quickly obtain important insights in audio collections.

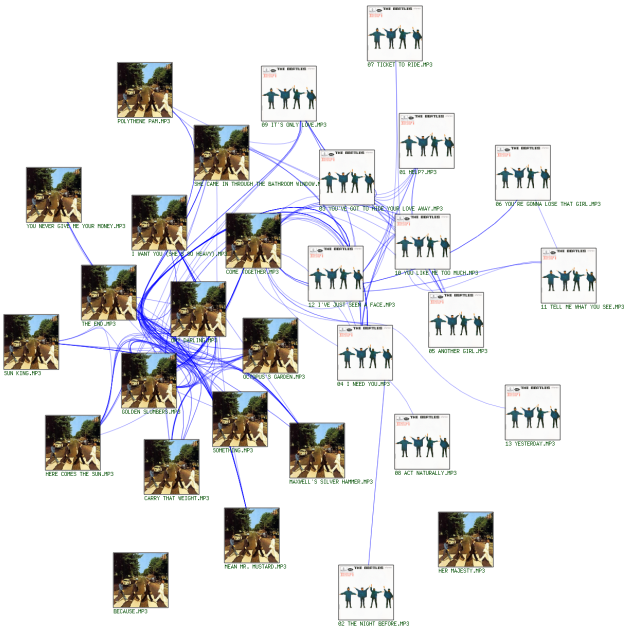
References

[1] Apple iTunes, http://www.apple.com/itunes/.

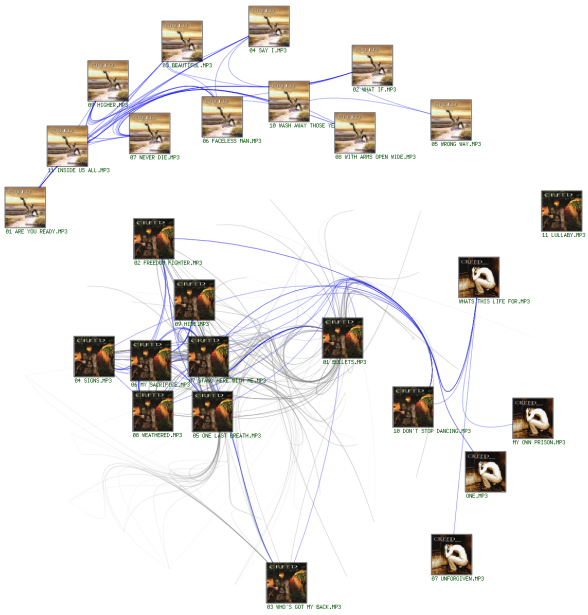
[2] J.-J. Aucouturier and F. Pachet. Music similarity measures: What's the use? In Ircam, editor, *Proceedings of the 3rd International Symposium on Music Information Retrieval*, pages 157–163, Paris, France, October 2002.

[3] Barcelona Music & Audio Technologies (BMAT). http://bmat.com/discover.

[4] A. Berenzweig, B. Logan, D. P. W. Ellis, and B. P. W. Whitman. A large-scale evaluation of acoustic and subjective music-similarity measures. *Comput. Music J.*, 28(2):63–76, 2004.

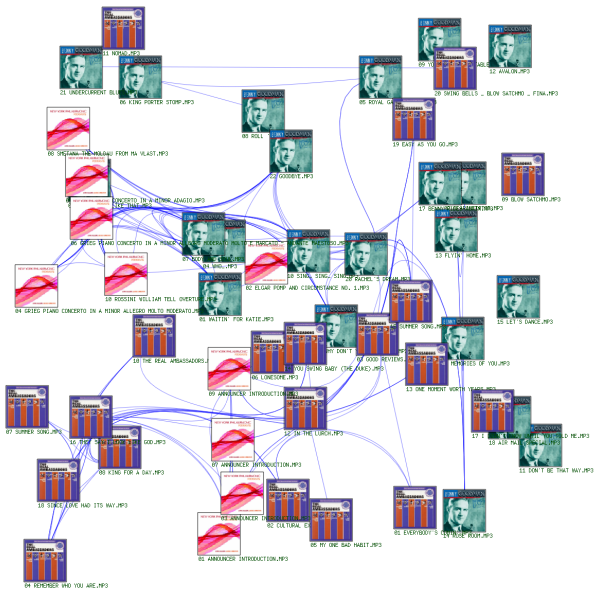


(a) *The Beatles* Our approach organizes songs by The Beatles according to album, while finding branches of similarity between them and identifying outliers.



(b) *Creed* Here, our approach splits the albums even more strongly, while finding connections between their 3rd and 1st albums, and many connections from their 3rd album out to the rest of the graph. Also, the lullaby is isolated completely, even though its tags say it is the same as the other songs.

(c) *Dr. Horrible.* The "Dr. Horrible" album contains songs with a variety of singers and a range of styles. As such, it is very spread out in the main graph, but the detail graph reveals that there are both similarities and differences. In particular, Felicia Day's solo songs with piano accompaniment are distinctly isolated in the lower left, as are the 'Bad Horse' songs in the upper left.



(d) *Jazz and Philharmonic* The system was used to search for Jazz and the Philharmonic. Interestingly, the two jazz albums are quite distinct, and the Philharmonic music splits between them. Looking at the details shows that one half of the Philharmonic tracks are performances and are linked to big-band swing music, while the other half of the Philharmonic tracks are actually spoken introductions to the songs, and are linked to more traditional Jazz.

Figure 4. *Case studies.* Exploring the graph though selection or searching reveals some interesting specifics of the music library and demonstrates the effectiveness of our approach.

[5] Y.-X. Chen and A. Butz. MusicSim: Integrating Audio Analysis and User Feedback in an Interactive Music Browsing UI. In *Proceedings of the 14th international conference on Intelligent User Interfaces, IUI 2009, Sanibel Island, Florida, USA, February 8-11, 2009*. ACM New York, NY, USA, Feb. 2009.

[6] J. Donaldson. Music recommendation mapping and interface based on structural network entropy. In *ICDEW '07: Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop*, pages 811–817, Washington, DC, USA, 2007. IEEE Computer Society.

[7] G. W. Furnas. Generalized fisheye views. In *CHI '86: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 16–23, New York, NY, USA, 1986. ACM.

[8] K. Hoashi, S. Hamawaki, H. Ishizaki, Y. Takishima, and J. Katto. Usability evaluation of visualization interfaces for content-based music retrieval systems. In *International Symposium on Music Information Retrieval, 2009*, Oct 2009.

[9] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12:741–748, 2006.

[10] P. Knees, M. Schedl, T. Pohle, and G. Widmer. An innovative three-dimensional user interface for exploring music collections enriched. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, pages 17–24, New York, NY, USA, 2006. ACM.

[11] P. Lamere and D. Eck. Using 3d visualizations to explore and discover music. In S. Dixon, D. Bainbridge, and R. Typke, editors, *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, pages 173–174, Vienna, Austria, September 2007. Österreichische Computer Gesellschaft.

[12] A. S. Lillie. Musicbox: Navigating the space of your music. Master's thesis, Massachusetts Institute of Technology, 2008.

[13] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[14] B. Logan and A. Salomon. A music similarity function based on signal analysis. *Multimedia and Expo, IEEE International Conference on*, 0:190, 2001.

[15] McKay, C., and I. Fujinaga. 2010. Improving automatic music classification performance by extracting features from different types of data. Proceedings of the ACM SIGMM International Conference on Multimedia Information Retrieval. 25766.

[16] C. Muelder and K.-L. Ma. Rapid graph layout using space filling curves. In *In Proceedings of IEEE Information Visualization Conference (InfoVis)*, 2008.

[17] C. Muelder, K.-L. Ma, and T. Bartoletti. A visualization methodology for characterization of network scans. In *ACM VizSEC 2005 Workshop*, pages 29–38, 2005.

[18] MusicLens, http://finetunes.musiclens.de/.

[19] musicmap. http://www.music-map.com/.

[20] Musicovery : interactive webradio. http://musicovery.com/.

[21] R. Neumayer, M. Dittenbach, and A. Rauber. Playsom and pocketsomplayer, alternative interfaces to large music collections. In *Queen Mary, University of London*, pages 618–623, 2005.

[22] A. Noack. An energy model for visual graph clustering. *Lecture Notes in Computer Science*, 2912:425–436, Mar. 2004.

[23] E. Pampalk. Islands of Music: Analysis, Organization, and Visualization of Music Archives. Master's thesis, Vienna University of Technology, Austria, December 2001. http://www.oefai.at/˜elias/music.

[24] E. Pampalk and M. Goto. Musicrainbow: A new user interface to discover artists using audio-based similarity and web-based labeling. In *Labeling, in the Proceedings of the ISMIR International Conference on Music Information Retrieval*, pages 367–370, 2006.

[25] E. Pampalk and M. Goto. Musicsun: A new approach to artist recommendation. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR '07)*, pages 101–104, Vienna, Austria, September 2007.

[26] E. Pampalk, A. Rauber, and D. Merkl. Content-based Organization and Visualization of Music Archives. In *Proceedings of the ACM Multimedia*, pages 570–579, Juan les Pins, France, December 1-6 2002. ACM.

[27] T. Phole, E. Pampalk, G. Widmer. "Generating Similarity-based Playlists Using Traveling Salesman Algorithms." Proc. of the 7th International Conference on Digital Audio Effects (Madrid, Spain, 2005). DAFx '05.

[28] L. Rabiner and B.-H. Juang. *Fundamentals of speech recognition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[29] L. Stefan and M. Topf. Globe of music - music library visualization using geosom. In *8th International Conference on Music Information Retrieval (ISMIR 2007)*, Vienna, Austria, 9 2007. sterreichische Computer Gesellschaft (OCG).

[30] TuneGlue Relationship Explorer. http://audiomap.tuneglue.net/.

[31] G. Tzanetakis, "Marsyas: a case study in implementing Music Information Retrieval Systems." Intelligent Music Information Systems: Tools and Methodologies 31-49. 2008.

[32] F. van Ham and A. Perer. "Search, Show Context, Expand on Demand": Supporting large graph exploration with degree-of-interest. *IEEE Transactions on Visualization and Computer Graphics*, 15:953–960, 2009.

[33] H. Vinet, P. Herrera, and F. Pachet. The cuidado project. In Ircam, editor, *Proceedings of the 3rd International Symposium on Music Information Retrieval*. Ircam, October 2002.