1 Introduction

Music Information Retrieval (MIR) tasks aim to learn about music, such as the structure of a piece, how various pieces compare to each other, and how to recommend music to listeners. There are many approaches to MIR tasks, and a single song can be represented in several different ways. In our project, we created an algorithm that detects structure in a musical song. By structure, we do not necessarily mean verse or chorus. Instead, our algorithm should detect the same kinds of structure that a human eye sees when looking at the matrix that represents a song.
Our goal is to find exact repeats of square-submatrices of varying sizes. A relaxed version of this goal is to cluster square-submatrices that are close to each other. Our algorithm is based on Ng et al's [4] spectral clustering algorithm. Our algorithm learns both the σ for creating the affinity matrix and the appropriate K, the number of clusters. Then our algorithm clusters the square-submatrices and performs various statistics on the resulting clusters.

2 Data Set

Our song data from Beatles Dataset created by Prof. Michael Casey, Department of Music, Dartmouth College. Each song is represented by distance matrix of squared-Euclidean distances between song‘s audio shingles. Audio shingles are comprised 30 concatenated feature vectors , created by splitting the audio track into tenth of a second windows and extracting the mel-frequency cepstral coefficients (MFCCs) for each of the window[1]. Song's Distance Matrix cleaned by separating recorded zeros from non-recorded distances with non-recorded distances set to 3. We chose 3 for these non-recorded values because the largest possible recorded value is 2. Thus by choosing 3, we have clearly separated the recorded values from the non-recoded ones. Below are two cleaned songs from the Beatles Dataset:

3 Method

Our algorithm has three phases: feature extraction, clustering, and evaluation. Our algorithm makes a number of assumptions. First, it has been shown that for this particular kind of song data that that repeats present themselves as diagonals[3]. Therefore we will only compare the diagonals of square-submatrices of the song data matrix. Second, we are fixing the size of the sub-matrrices to be 100×100. We have also decided to comparison of diagonals whose sub-square-matrices overlap in someway. Lastly due to memory constraints, we will only consider one band of diagonals. We will only be considering diagonals begin at entry (1,i) for all i from 1 to (n-100+1), where n is the number of rows (and columns) of the full song matrix.
Using the above assumptions, we rewrite our goal as follows: we group square-submatrices together if their diagonals are identical. Thus the relaxed version is that we group square-submatrices together if their diagonals are near each other (i.e. if they are in the same cluster.

3.1 Algorithm – Learning σ and K

1. Feature Extraction
   In this step, we will extract one band of diagonals and convert the diagonals into columns (to be acceptable input for MATLAB). Then we will create a pairwise squared-Euclidean distance matrix $\mathcal{D}$.

2. Perform Spectral Clustering (based on algorithm from [4])
   In this step of the algorithm, we first need to find an appropriate value for σ that will be used to create the affinity matrix A, which is defined as $A_{ij} = \exp\left(\frac{-\mathcal{D}_{ij}}{2\sigma^2}\right)$ for $i \neq j$ and $A_{ii} = 0$. Given a vector of possible values for σ, our algorithm looks for the value that minimizes the mean of the entries of A, while also looking for the sigma which maximizes the value for the variance. In our algorithm, we compute A for each of the possible σ values, then choose the five values that give the lowest mean for the entries of A. Then we choose of those five possible σ the one whose associated A has the largest variance. (Although we did run an experiment of first choosing five values of σ based on the largest variance and then chose the value of σ that yields the A with the smallest mean. Our resulting σ was the same for each experiment. This of course makes sense to us due to definition of variance and the fact that the entries of A are all between 0 and 1. Thus if we subtract a small mean from each of the entries of A - as one does in the computation of variance - then we would expect the mean of the entries of the resulting matrix to be higher than if we had subtracted a larger mean from each of the entries of A).

   Using this affinity matrix, we compute the eigenvalues using SVD. We note that since $\mathcal{D}$ is symmetric, then so is A. Thus the eigenvalues of A are real. As shown in class, if we define Λ to be the diagonal matrix with the eigenvalues of A along the diagonal in descending order and if we define U to be the matrix comprised of the eigenvectors of A as its columns, then we have that:

$$\begin{aligned} AU &= \Lambda U = U\Lambda \\ AUU^t &= U\Lambda U^t \\ A &= U\Lambda U^t \end{aligned}$$

   Therefore, we use the preferred MATLAB function SVD() instead of EIG().

   The next step of the algorithm requires choosing K, the number of clusters. We have two methods for making this choice. In the first method, we approximate the tangent curve of the relative eigenvalues and choose the first K where this approximated tangent curve has slope within ϵ_λ of 0. We approximate the tangent curve at the K^{th} relative eigenvalue difference by using the K^{th} and $(K-1)^{\text{th}}$ relative eigenvalue differences. Similarly for method two, we attempt to approximate the tangent curve of the eigenvalues and choose the first K where this approximated tangent curve has slope within ϵ_λ of 0. However to approximate the tangent curve at K we use the $(K+1)^{\text{th}}$ and $(K-1)^{\text{th}}$ eigenvalues. We are aware of the inconsistency between the two methods.

but we note that in the second method, we know that the eigenvalues are in descending order, while for the first method, we do not have such knowledge. In fact, we note that it is possible to have the relative eigenvalue differenced fluctuating between positive and negative results, hence the use of two consecutive relative eigenvalue differences, while for the eigenvalue differences we know that they are all negative. Also, please note that for our experiments, we chose $\epsilon_\lambda = 0.01$. We then use this K to cluster the diagonals of the square-submatrices.

3. Evaluation In the final step of our algorithm, we determine if the clusters are comprised of near-repeats. Noting that if a cluster is comprised of diagonals that are all close to each other, then for any two diagonals $d, \widetilde{d}$, we have that $\|d - \widetilde{d}\| \leq \epsilon$ for a small value of $\epsilon > 0$. Therefore, we will evaluate each of our clusters by computing the range, mean, and variance of the pairwise distances between distinct diagonals belonging to that cluster.

4 Results & Discussion

For both songs from the data set, we found $\sigma = 0.05$. Our method is flawed - for both songs and both methods for finding K, there is one cluster has range of pairwise distances significantly larger than 0. We note that we found a typo in our method 1 computation for K and the below results for those methods are different than the presented poster.

As a result of the discussion at the poster session, I created three test matrices. The first (SongID = 3) is created by tiling the row $A = [0, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 3]$ to become a matrix of size 698×698. The second (SongID = 4) is created by tiling the 10×10 identity matrix to be a matrix of size 698×698. The third one is the first 10 columns tiled to create a matrix that is 698×698. In all three cases, we expect there to be 10 clusters. Unfortunately, in none of these cases did this happen. In fact, we had to further the code to handle empty clusters due to these cases. Please see Figure 5 for the visualizations.

After looking at the results for all three test matrices, we certainly feel that the method must be flawed because for none of the songs in either method of computing K does the algorithm find 10 clusters as we would expect. For further comparison, we ran k-means on these test matrices using $k = 10$, the K that the algorithm found before performing k-means (K_B) and the K that the algorithm found after performing k-means (K_A). Below is the number of clusters that k-means found performed on the extracted diagonals from the test matrix (before the spectral clustering step).

SongID	Method	K = 10	K_B	K_A
3	1	10	2	2
3	2	10	2	2
4	1	10	4	4
4	2	10	4	2
5	1	10	2	2
5	2	10	2	2

We notice that if we set $K = 10$ for the test matrices and then cluster the resulting diagonals, we still have 10 diagonals. Thus we believe that there must be a flaw in either our method for choosing σ or our method for choosing K. We should also note that the second test matrix (Song ID = 4) for the second method for finding K, the later k-means computation resulted in two empty clusters and the number of clusters was therefore reduced.

5 Future Work

We recognize that our work is just a first step into building a hierarchical decomposition for song data. But even before building such a decomposition, some future work may include perform network analysis on resulting clusters with largest range, creating an algorithm that learns the appropriate band width, and creating an algorithm that ignores diagonals that have no near-repeats (i.e. ones that are too far from all other diagonals). After building a reliable algorithm for this first decomposition step, then we could use this technique recursively on resulting clusters to create hierarchical representations for our song data

6 References

[1] M. Casey, C. Rhodes, and M. Slaney, *Analysis of minimum distances in high-dimensional musical spaces*, IEEE Transactions on Audio, Speech, and Language Processing **16** (2008), no. 5, 1015 − 1028.
[2]M. Meila and L. Xu *Multiway Cuts and Spectral Clustering. Retrieved from: http://www.stat.washington.edu/mmp/Papers/*
[3]M. Müller, P, Grosche, and N. Jiang, *A Segment-based fitness measure for capturing repetitive structures of music recordings*, 12th International Society for Music Information Retrieval Conference (ISMIR 2011), Miami, Oct, 2011.
[4]A. Y. Ng, M. I. Jordan, Y. Weiss, *On Spectral Clustering: Analysis and an algorithm*, Advances in Neural Information Processing Systems 14: Proceedings of the 2002 conference **2** (2002), 849 − 856. *Retrieved from: http://ai.stanford.edu/ ang/papers/nips01-spectral.pdf*
[5]J. Shi and J. Malik, *Normalized Cuts and Image Segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence **22** (2000), no. 8, 888 − 905
[6]U. von Luxburg, *A Tutorial on Spectral Clustering*, Mathematics and Computing **17** (2007) no. 4, 395 − 416. *Retrieved from: http://www.kyb.mpg.de/fileadmin/user_upload/files/publications/attachments/Luxburg07_tutorial_4488%5b0%5d.pdf*

7 Images

Below are the resulting images for our algorithm. First are the images for the cleaned songs, which are followed by the test matrices.

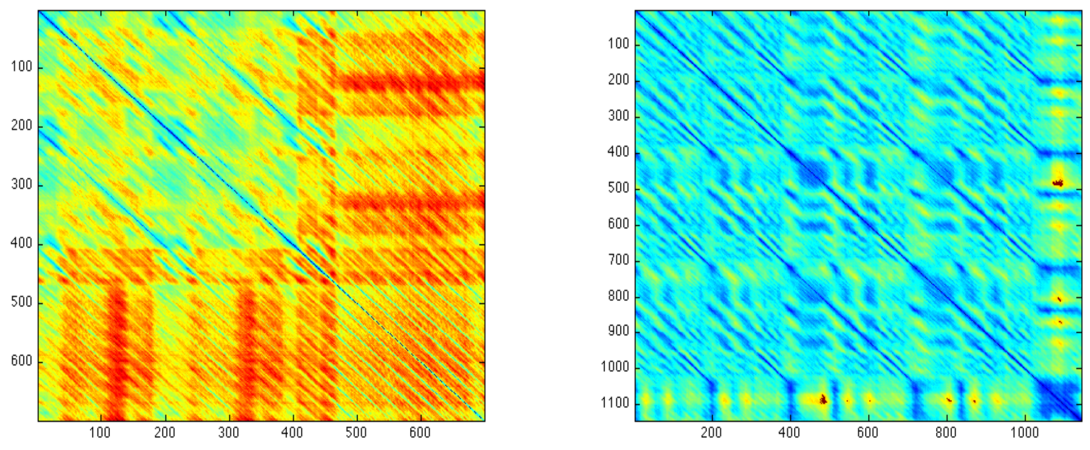

Figure 1: Cleaned song visualizations for “Polythene Pam” (Abbey Road), “You Know What To Do” (Anthology 1, Disc 2)

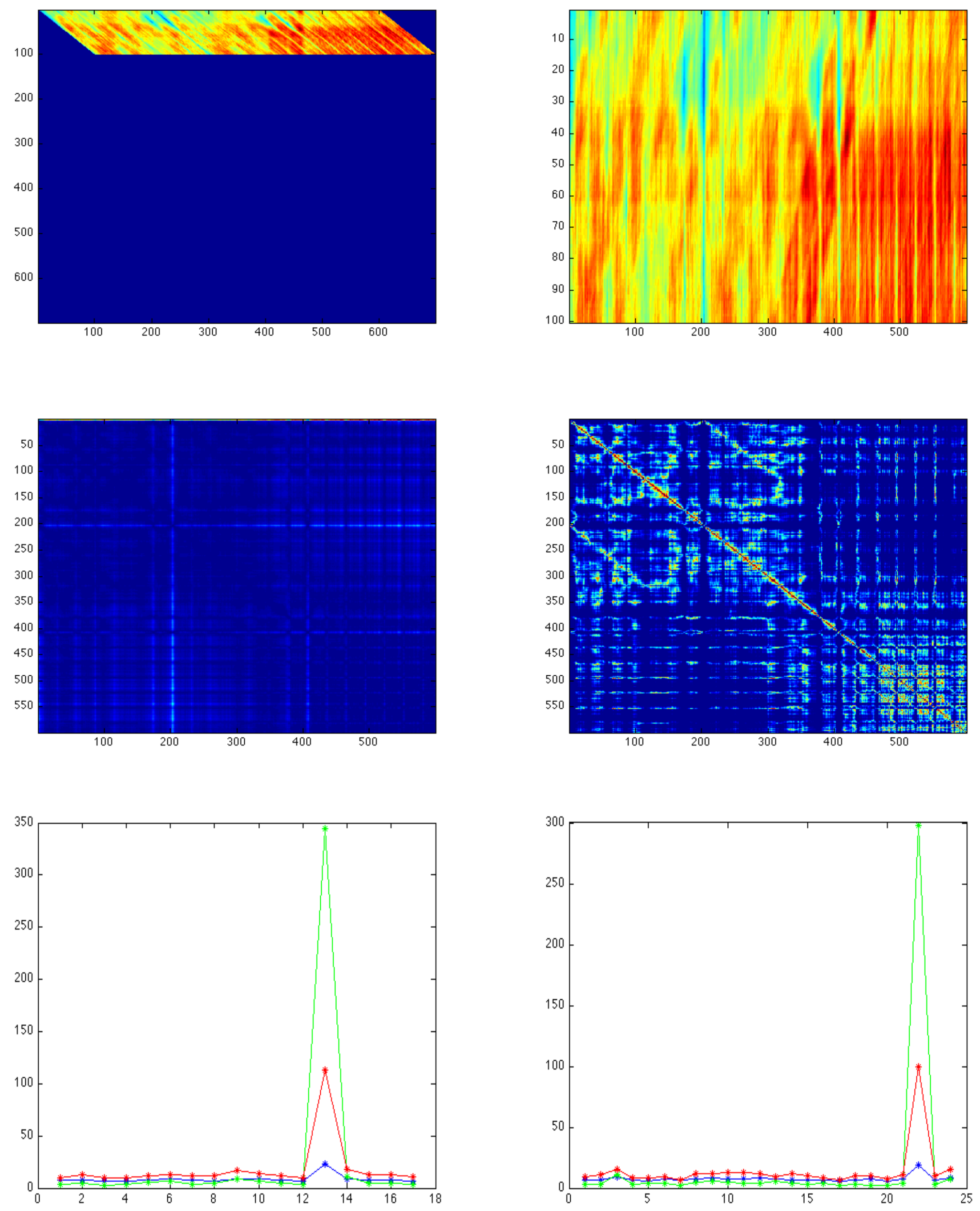

Figure 2: “Polythene Pam” Isolated Diagonals - (a) with remaining cleaned song set to 0, (b) as columns, (c) normalized pairwise distances between diagonals, (d) resulting affinity matrix, (e)-(f) plots of Range (red), Mean (blue), Variance (green) for $\mathcal{C}^k$ for - (e) Method 1: K = 17, (f) Method 2: K = 24

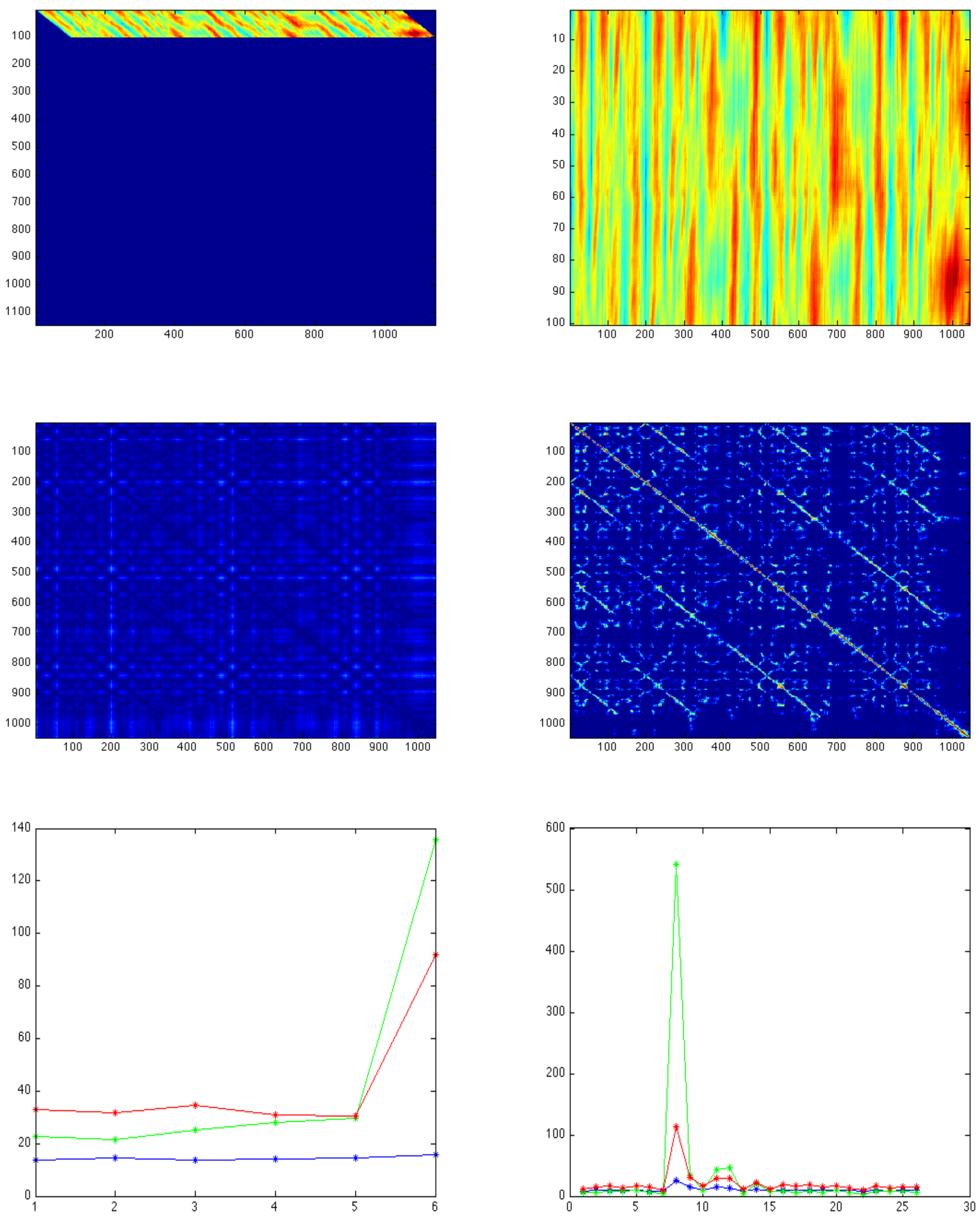

Figure 3: “You Know What To Do” Isolated Diagonals - (a) with remaining cleaned song set to 0, (b) as columns, (c) normalized pairwise distances between diagonals, (d) resulting affinity matrix, (e)-(f) plots of Range (red), Mean (blue), Variance (green) for $\mathcal{C}^k$ for - Range (red), Mean (blue), Variance (green) for $\mathcal{C}^k$ - (b) Method 1: $K = 6$, (c) Method 2: $K = 26$

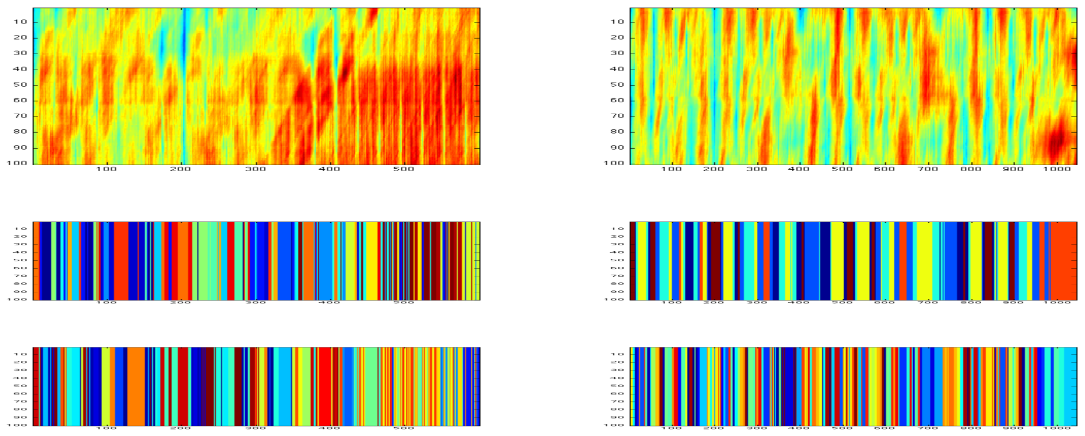

Figure 4: “Polythene Pam” - (a) Diagonals as columns, Clustering assignments visualizations: (c) Method 1, (e) Method 2. Similarly (b),(d),(f) for “You Know What To Do”

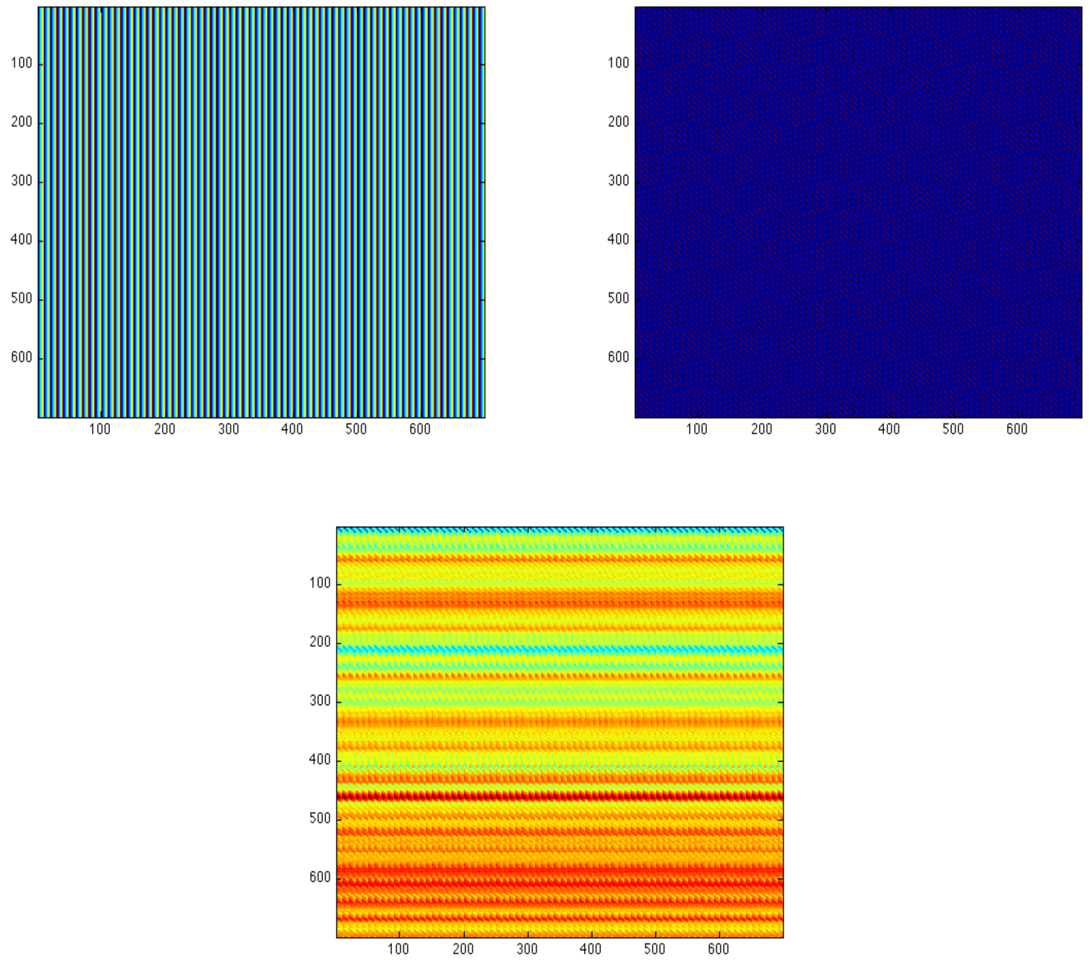

Figure 5: Visualizations of the three test songs (SongID 3, 4, 5)

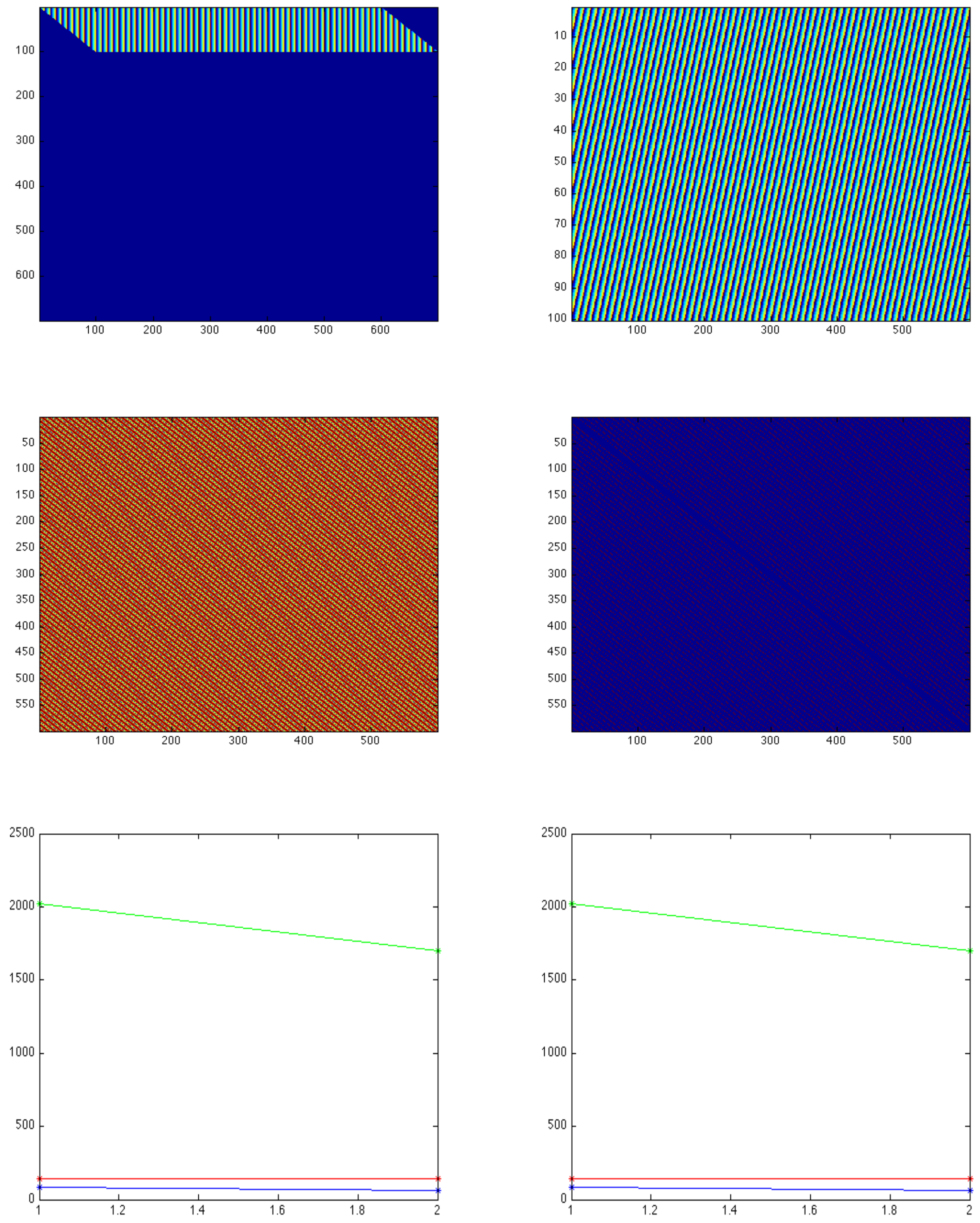

Figure 6: First Test Matrix (SongID = 3) Isolated Diagonals - (a) with remaining cleaned song set to 0, (b) as columns, (c) normalized pairwise distances between diagonals, (d) resulting affinity matrix, (e)-(f) plots of Range (red), Mean (blue), Variance (green) for $\mathcal{C}^k$ for - Range (red), Mean (blue), Variance (green) for $\mathcal{C}^k$ - (b) Method 1: $K = 2$, (c) Method 2: $K = 2$

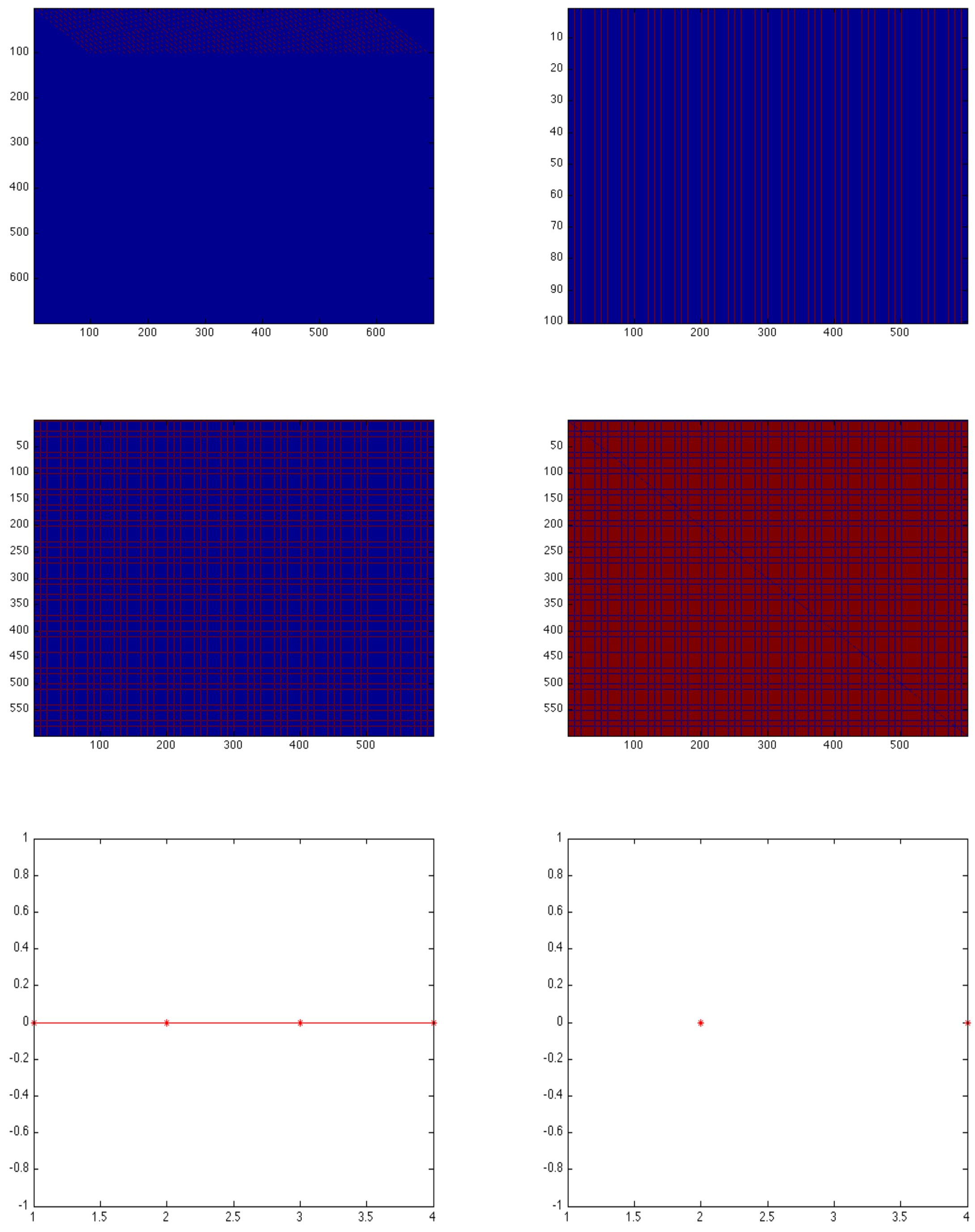

Figure 7: Second Test Matrix (SongID = 4) Isolated Diagonals - (a) with remaining cleaned song set to 0, (b) as columns, (c) normalized pairwise distances between diagonals, (d) resulting affinity matrix, (e)-(f) plots of Range (red), Mean (blue), Variance (green) for $\mathcal{C}^k$ for - Range (red), Mean (blue), Variance (green) for $\mathcal{C}^k$ - (b) Method 1: $K = 4$, (c) Method 2: $K = 2$ (original found K = 4)

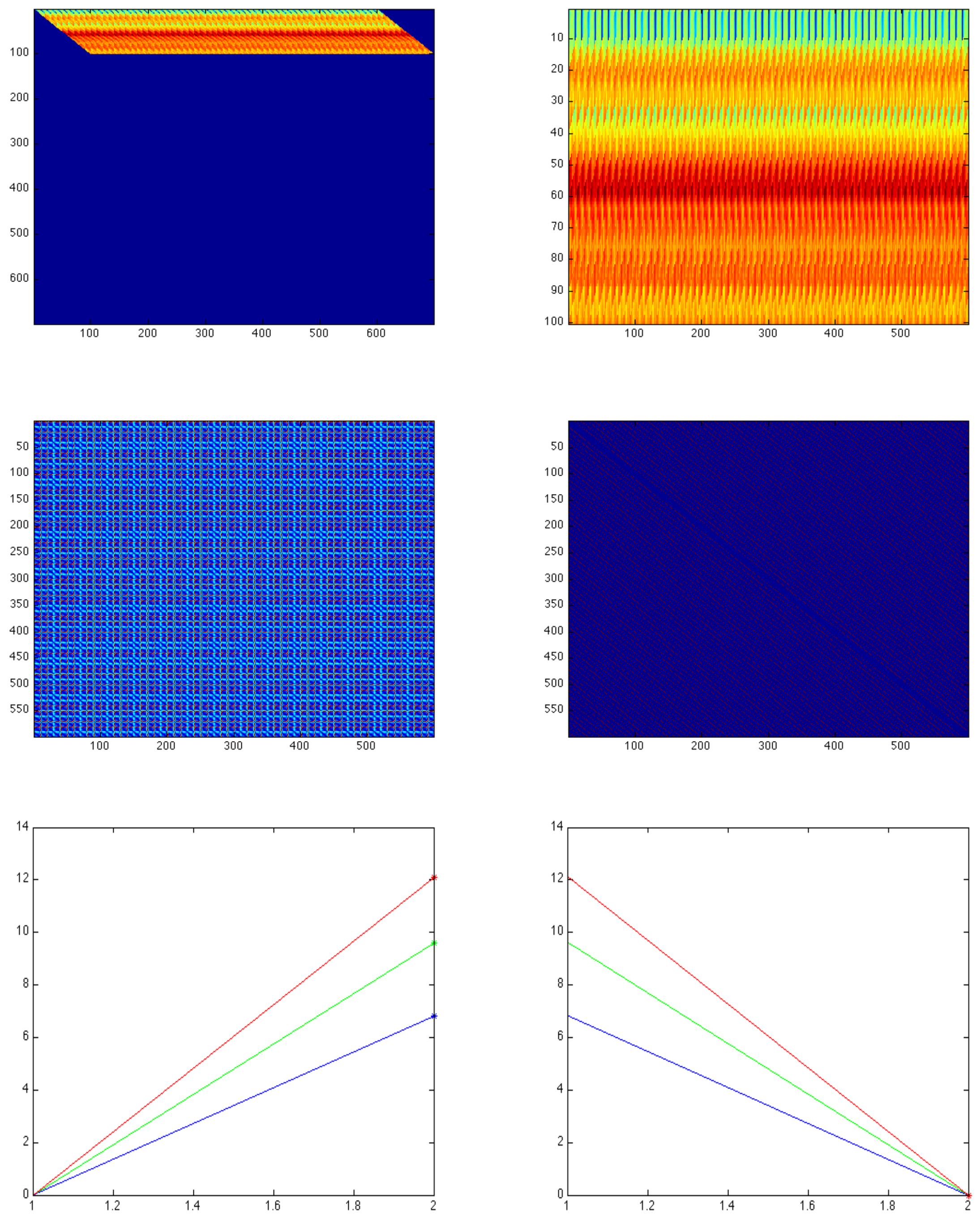

Figure 8: First Test Matrix (SongID = 3) Isolated Diagonals - (a) with remaining cleaned song set to 0, (b) as columns, (c) normalized pairwise distances between diagonals, (d) resulting affinity matrix, (e)-(f) plots of Range (red), Mean (blue), Variance (green) for $\mathcal{C}^k$ for - Range (red), Mean (blue), Variance (green) for $\mathcal{C}^k$ - (b) Method 1: $K = 2$, (c) Method 2: $K = 2$

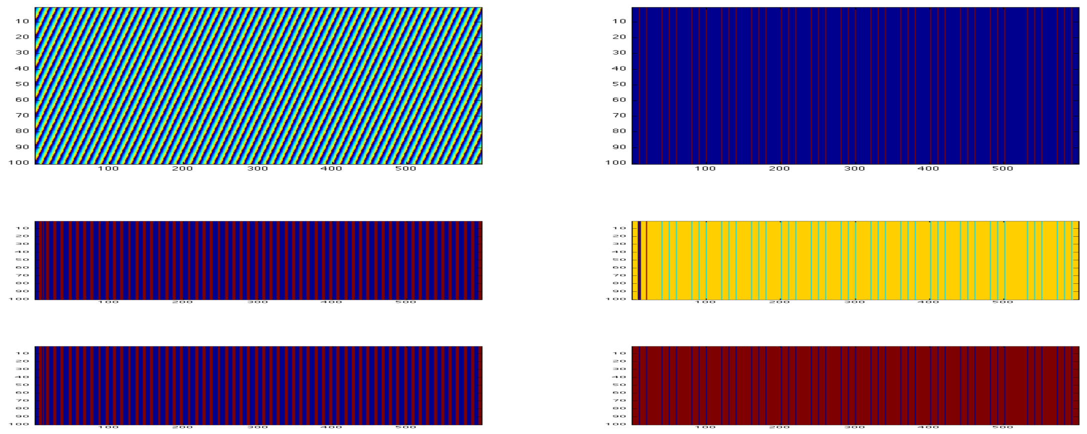

Figure 9: SongID = 3 - (a) Diagonals as columns, Clustering assignments visualizations: (c) Method 1, (e) Method 2. Similarly (b),(d),(f) for Song ID = 4

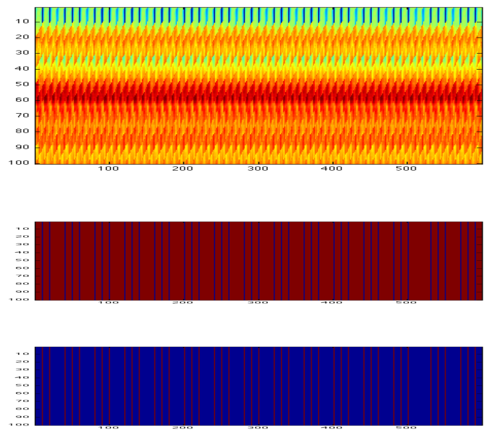

Figure 10: SongID = 5 - (a) Diagonals as columns, Clustering assignments visualizations: (b) Method 1, (c) Method 2.