Classification of Time Series Data (e.g. Music)

Kellen Betts

Updated July 31, 2016

Abstract

In this project an algorithm is developed to classify music genre and bands based on their time-frequency signatures. An assortment of music from my iTunes collection is used to build training sets for band classification of bands from (1) different genres and (2) the same genre, and (3) classification of different genres with assorted bands. The computational methods used include time-frequency analysis with the Gábor transform, singular value decomposition (SVD) to identify dominant modes in the time-frequency data, and linear discriminant analysis (LDA) to provide the classification metric. The results show that this algorithm is most effective with band classification from different genres with over an 80% accuracy rate. Classification of different genres and bands within a single genre show less promising results with less than 70% accuracy rates. The number of time-frequency signatures (identified as modes from the SVD) necessary for optimal classification is extensively explored with approximately ten modes proving most effective.

1 Introduction

The objective in this project is to develop an algorithm that can distinguish between music bands and genres. The basic components include time-frequency analysis using the Gábor transform, singular value decomposition (SVD) to identify dominant modes in the time-frequency data, and linear discriminant analysis (LDA) to provide a classification metric. An assortment of data from my iTunes collection is used to build training sets and test the efficacy of the algorithm. The test conditions used for this analysis were:

1. **Test 1** (Band Classification I): Three different bands from *different* genres. The bands used are Bob Marley (reggae), The Offspring (modern or alternative rock), and Alireza Eftekhari (perhaps best classified as "international").

2. **Test 2** (Band Classification II): Three different bands from the *same* genre making separation much more difficult. The genre used is modern or alternative rock and the bands used are Phish, Coldplay, and Dave Matthews Band.

3. **Test 3** (Genre Classification): An assortment of bands from three different genres are used to classify tracks by genre. The genres used are classical, techno or electronica, and modern or alternative rock with various bands

2 Theoretical Background

2.1 Time-Frequency Analysis (Gábor)

The basis used for time-frequency analysis in this project is the Gábor transform. This technique is a windowed Fourier transform where time and frequency components are identified. For a given function

$f(x)$ the Fourier transform and its inverse are defined,

$$F(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ikx} f(x)\, dx \tag{1}$$

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{ikx} F(k) dk \tag{2}$$

where k corresponds to the wave-numbers in the trigonometric identity. With the Fourier transform, the spectral content can be effectively isolated from a signal but all temporal information is lost because the signal is integrated over the domain $x\,\epsilon\,[-\infty, \infty]$. The Gábor transform extends this technique by defining a time window $g(\tau - t)$ that allows the transform to localize both time and frequency content. For a given function $f(t)$ the Gábor transform is defined (Kutz 12.4.1),

$$G[f](t, \omega) = \int_{-\infty}^{\infty} f(\tau)\, \bar{g}(\tau - t)\, e^{-i\omega\tau} d\tau\,.$$

Computationally the Gábor transform is implemented over the finite domains (Kutz 12.4.9a-b)

$$\tau = b\, t_0 \tag{3}$$

$$v = a\, \omega_0 \tag{4}$$

and so the discrete kernel for the transform becomes (Kutz 12.4.10),

$$g_{m,n}(t) = e^{i2\pi a \omega_0 t} g(t - b\, t_0)\ .$$

Frequency content is isolated using the Fast Fourier Transform (FFT) algorithm which can achieve a low operation count of $O(N \log N)$ assuming a 2^n discritization. Additionally, given its trigonometric construction, the Fourier transform assumes a 2π-periodic domain.

To filter the audio tracks, the first Gábor function used is a Gaussian (Kutz 12.7.54),

$$g(t) = e^{-a(t-b)^2} \tag{5}$$

where a is the window width and b is the length of the temporal shift. The Gaussian is then compared to the Mexican Hat function defined by (Kutz 12.5.24a),

$$\psi(t) = \left(1 - t^2\right) \exp\left(-\frac{t^2}{2}\right)$$

which is implemented using the mother wavelet (Kutz 12.5.14),

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\, \psi\left(\frac{t-b}{a}\right) \tag{6}$$

and so the function for the Mexican Hat Wavlet is given by,

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}} \left(1 - \left(\frac{t-b}{a}\right)^2\right) \exp\left(-\frac{\left(\frac{t-b}{a}\right)^2}{2}\right). \tag{7}$$

2.2 Singular Value Decomposition

The singular value decomposition (SVD) is defined,

$$X = U\Sigma V^* \tag{8}$$

where Σ is a diagonal matrix of singular values σ that correspond to the square root of the eigenvalues $\left(\sigma_i = \sqrt{\lambda_i}\right)$ and are ordered based on size. U and V^* are orthonormal unitary transformations corresponding to the orientation of the incoming data. Using the proper orthogonal modes from the SVD, a low rank approximation can be calculated by the sum,

$$X_N \approx \sum_{j=1}^{N} \sigma_j \vec{u}_j \vec{v^*}_j \tag{9}$$

where N is a given rank. The 2-norm error for the approximation is given by,

$$\|X - X_N\|_2 = \sigma_{n+1}\,. \tag{10}$$

One way to think about SVD is as a least squares fit in higher dimensional space where the elements of Σ are ordered by their L^2 energy. The energy for a given mode can be calculated,

$$\text{energy}_i = \frac{\sum_{i=1}^{N} \sigma_i}{\sum_{i=1}^{n} \sigma_i}\,. \tag{11}$$

Therefore, diagonalization with SVD is a transformation to a new orthonormal basis and provides low rank approximations for a given dataset.

2.3 Linear Discriminate Analysis

The central idea behind LDA is to "find a suitable projection that minimizes the distance between the inter-class data while minimizing the intra-class data" (Kutz p.378). This is achieved by the LDA projector (w),

$$w = \arg\max_w \frac{w^T S_B w}{w^T S_W w} \tag{12}$$

The sample covariance of the class means is given by the inter-class scatter matrix (S_B) defined by (Wikipedia, http://en.wikipedia.org/wiki/Linear_discriminant_analysis),

$$S_B = \sum_{i=1}^{C} (\mu_i - \mu)(\mu_i - \mu)^T \tag{13}$$

where C is the number of classes (in this case 3), μ_i is the mean for a given classes, and μ is the mean of the entire dataset. The intra-class scatter matrix (S_W) is defined by,

$$S_W = \sum_{j=1}^{C} \sum_{x} (x - \mu_j)(x - \mu_j)^T \tag{14}$$

For this analysis, the classes are the three bands or genres being classified. The LDA projection vector is found by obtaining the eigenvector of the maximum eigenvalue in the eigenvalue problem,

$$S_B w = \lambda S_W w\,. \tag{15}$$

3 Algorithm Implementation and Development

The key steps in the algorithm are:

1. Build training data set:
   (a) Import music tracks
   (b) Gabor transform
   (c) SVD and LDA to provide test metric
2. Test new tracks:
   (a) Import
   (b) Gabor transform
   (c) SVD and LDA projection
   (d) Classify results

The music used for this project was extracted from my iTunes music collection. All of the tracks were preprocessed in an audio editing application (*Audacity*, http://audacity.sourceforge.net/). Approximately 50 minutes of music for a given group (say Dave Matthews Band or a mix of "Classical") were imported into a single file, converted to a mono track (single channel), resampled at 8000 Hz, and exported as a .wav file. In MATLAB loading and resampling of the data is done in a subroutine (`loadTrack`) where the full .wav file is read in and 5 second samples are extracted randomly to prevent bias in sampling. Random selection is implemented by using a pseudo-random integer as the start point for extracting a 5 second sample. Once imported, the sampling frequency for each track is reduced to 2000 Hz (factor of 4) to reduce the size of the data and allow more samples. Ideally, a large number of samples in each group would be used for the training set. After exploring the parameters for this algorithm, the optimal balance between reducing the sampling frequency, increasing the number of samples, and preventing memory overload with the SVD operation is 20 samples for each group meaning the full training set has 60 tracks.

Once the tracks for the training set are imported, time-frequency analysis is done using the Gábor transform algorithm developed for Homework #2. The algorithm is run in a subroutine (`gabor.m`) that processes a single track input, so a loop is run over the full training set. The temporal domain is discritized with $n+1$ points then trimmed to n points since the FFT assumes periodic boundaries. The frequency domain is discritized using,

$$\mathtt{k} = \frac{2\pi}{\mathtt{L}}\left[\mathtt{0} : (\frac{\mathtt{n}}{\mathtt{2}} - \mathtt{1}) - \frac{\mathtt{n}}{\mathtt{2}} : -\mathtt{1}\right] \tag{16}$$

which aligns with the shifted and 2π-scaled domain of the FFT algorithm. The Gaussian and Mexican Hat filter windows are implemented as autonomous functions,

$$\mathtt{gaussian} = @(\mathtt{a},\mathtt{b})\mathtt{exp}(-\mathtt{a} * (\mathtt{t} - \mathtt{b}).^{\mathtt{2}}) \tag{17}$$

$$\mathtt{mexHat} = @(\mathtt{a},\mathtt{b})(\mathtt{1}/\mathtt{sqrt}(\mathtt{a})) * (\mathtt{1} - ((\mathtt{t} - \mathtt{b})/\mathtt{a}).^{\mathtt{2}}). * \mathtt{exp}(-(((\mathtt{t} - \mathtt{b})/\mathtt{a}).^{\mathtt{2}})/\mathtt{2}) \tag{18}$$

Testing was done with the two filters and better results were obtained with the Gaussian so it was selected for the analysis. Temporal sampling is implemented using a linear vector that spans the length of the temporal domain and is discritized to the sampling parameter (b),

$$\mathtt{slide} = \mathtt{linspace}(\mathtt{t(1)}, \mathtt{t(end)}, \mathtt{b})\,. \tag{19}$$

The algorithm uses a loop for the temporal sampling in which the filter function is applied to the track vector then transformed using the FFT. Experimentation with the filter width (a) and temporal

sampling (b) was done to determine optimal values, and $a = 500$, $b = 100$ were selected for the analysis. The data returned by the `gabor.m` subroutine is shift from the FFT domain and return in as a row vector.

After the full training set is processed using the Gábor transform, the data is then sent to a subroutine (`trainer.m`) that implements the SVD and linear discriminant analysis. A reduced SVD is computed using the built-in MATLAB function which returns the U, S, and V matrices. The principal components are calculated for a given number of "features" or modes. (The orientation of the training data that was more efficient for the SVD function meant the principal components were calculated SU^T.) From the principal components, means are calculated for each group and the scatter matrices S_b and S_w are built. Next the eigenvalue problem (equation 15) is solved using the built-in MATLAB function `eig` and the maximum eigenvalue and corresponding eignevector (LDA projector w) are found. Finally, the principal components for each group in the training set are projected on the LDA projector (w) and means for each group are calculated. The `trainer.m` subroutine then returns the results from both the SVD and LDA for further analysis.

The tracks that are used to test the LDA training set are imported and processed by the Gábor transform in the same manner as the training data. This analysis used 10 tracks from each group (30 tracks total) for testing. The resulting time-frequency data is then projected onto the SVD components using the V^T obtained from the SVD and subsequent LDA projector (w) from the training set. Classification of a selected track is then done by comparing the projection values to the groups means from the train set with the closest value used.

Exploration of the "features" (modes from the SVD/LDA) is done by running the with 1, 2, 4, 6, 8, 10, 20, 30, 40, 50, and 60 (full rank) features. Accuracy rates are averaged over 9 trials for each value of the parameter to get a better estimate of the performance of the algorithm given the pseudo-random sampling procedure. The accuracy rate is calculated as a percent correct where 100% perfect performance. Means and standard deviations are reported and discussed in Section 4 (Computational Results), and the full data set is available in Appendix D.

4 Computational Results

4.1 Test 1

The objective of this test is to build a train set for three different bands from three different genres of music. The test is to see if the algorithm can classify new songs from the bands correctly. The three bands used are Bob Marley (reggae), The Offspring (modern or alternative rock), and Alireza Eftekhari (perhaps best classified as "international"). As seen in Figure 1, the original audio data from these three groups have distinct audio signatures. SVD of the time-frequency content shows (Figure 2) that the first mode is dominant but there is a clear heavy-tail distribution of the singular values. Spectrograms of the first four principal components (Figure 3) shows distinct time-frequency signatures for each component. Projection of the individual tracks onto the first three POD modes (Figure 4) shows shows high intra-band variance and distinct signatures between the groups**.** Figure 5 shows histograms of the projections confirming the high intra-band variance and distinct signatures for each band.

The results are summarized in Table 1. The averages (over 9 trials) indicate that the algorithm is near 80% accuracy for almost all modes. Peak performance at over 80% accuracy is seen with the 6, 8, and 10 rank projections. High deviations (particularly with higher rank projections) confirming the variability within a given group as seen in the projections (Figure 4). Since pseudo-random sampling is used to extract the track data it is possible that some of the samples may have contain little or no "musical" content (i.e. sampling at the beginning or end of a song). Nonetheless, these results strongly suggest that the algorithm was able to achieve over 80% accuracy for band classification with low rank approximations.

# of Features	Accuracy (Mean)	Std. Dev.
1	77%	7.5%
2	66%	11.8%
4	79%	10.3%
6	82%	9.9%
8	81%	10.6%
10	83%	7.2%
20	76%	15.8%
30	75%	11.9%
40	74%	13.0%
50	68%	16.2%
60	58%	12.0%

Table 1. For Test 1, mean and standard deviation results from 9 trials at the given values of the "features" parameter." The algorithm has an over 80% accuracy rate for classification of bands in different genres.

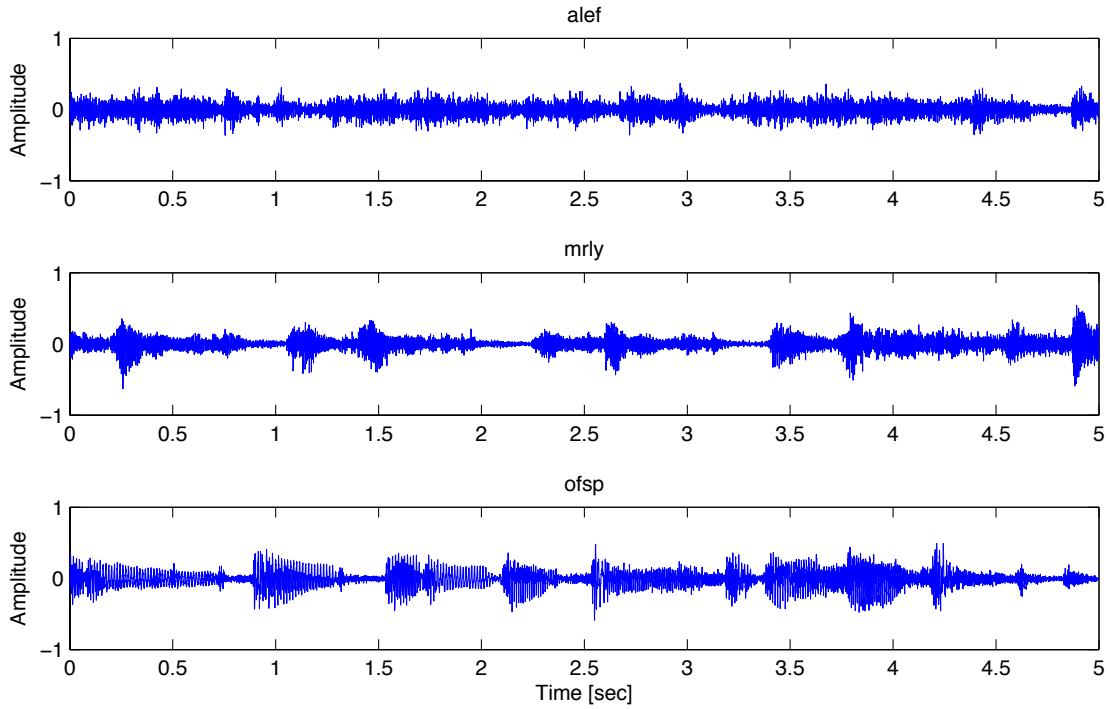

Figure 1. For Test 1, samples of the original audio show that the three groups Alireza Eftekhari (top), Bob Marley (center), and The Offspring (bottom) have distinct audio signatures.

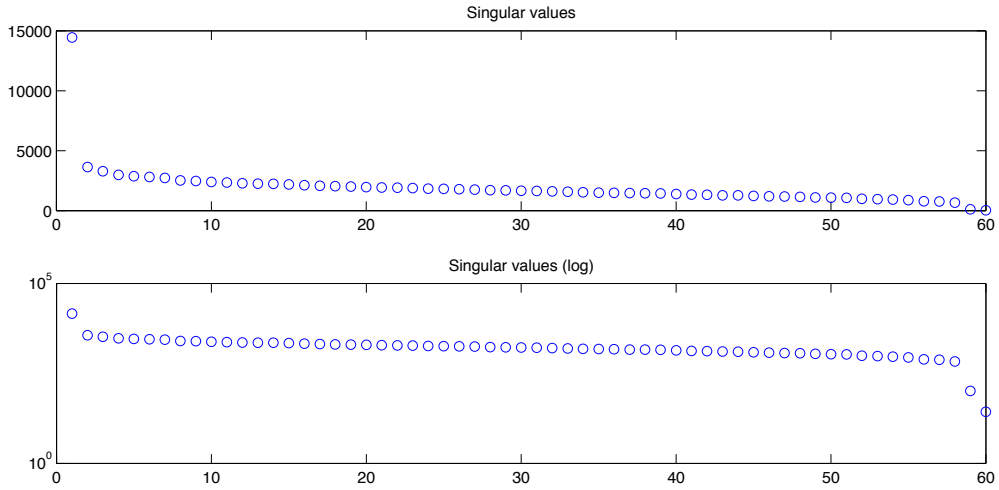

Figure 2. For Test 1, the first singular value is dominant, but there is a clear heavy-tail distribution.

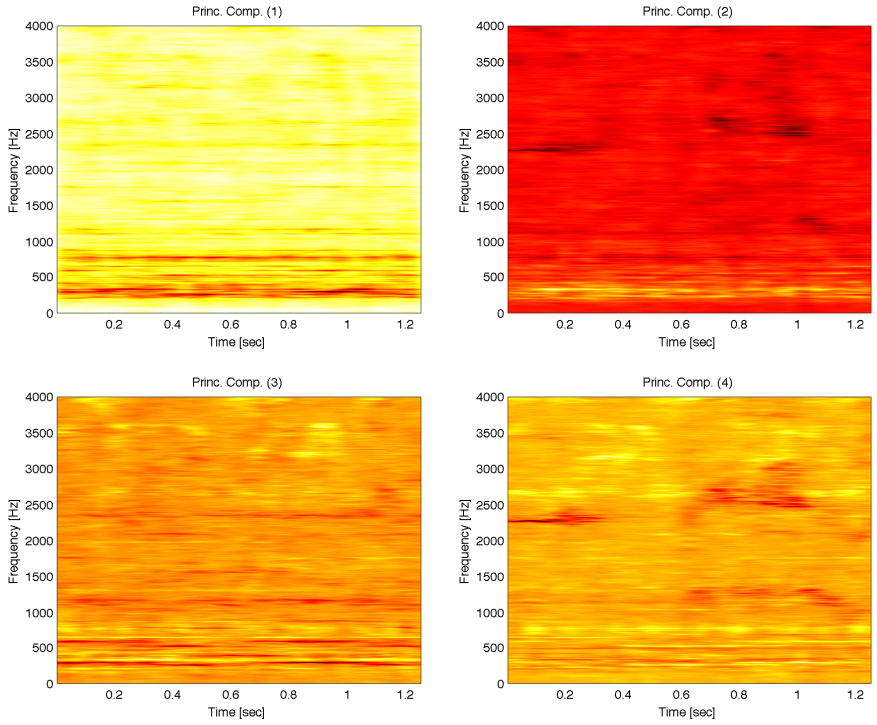

Figure 3. For Test 1, the first four principal components show distinct time-frequency signatures.

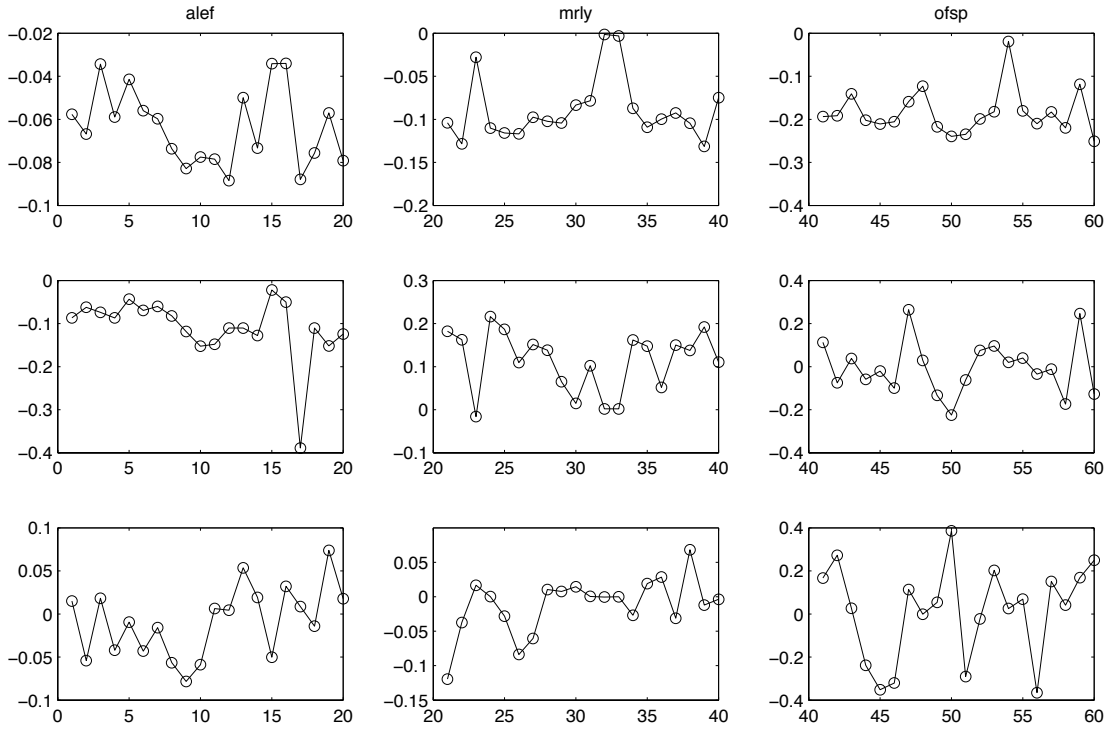

Figure 4. For Test 1, projection of the individual tracks onto the first three POD modes shows high intra-band variance and distinct signatures for each group: Alireza Eftekhari (left), Bob Marley (center), and The Offspring (right).

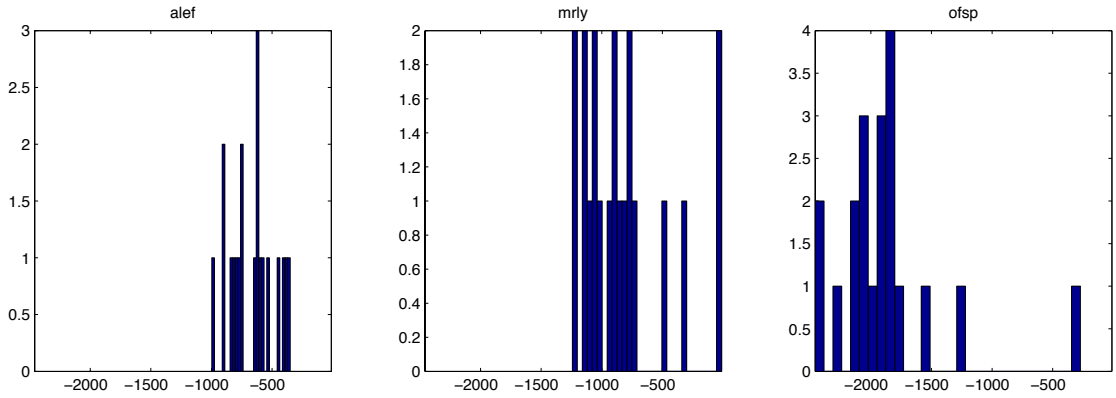

Figure 5. For Test 1, histograms of the projections shows high intra-band variance and distinct signatures for each band: Alireza Eftekhari (left), Bob Marley (center), and The Offspring (right).

4.2 Test 2

The objective of this test is to classify bands within a single genre (modern or alternative rock). This is a much more difficult test condition because within a genre there is much less variance between the bands than the first test condition. The music tested is from Phish, Coldplay, and Dave Mathews Band. As seen in Figure 6, the samples of the original audio is very similar for both Phish and Dave Matthews, and the track from Coldplay is the only one with notable visual difference. As with the first test condition there is a clear heavy-tail distribution of the singular values (Figure 7) but the first mode is still dominant. Spectrograms of the first four principal components (Figure 8) show distinct time-frequency signatures for each component. Projection of the individual tracks onto the first three POD modes (Figure 9) shows distinct signatures for each group and high intra-band variance. The histograms in Figure 10 confirms the high intra-band variance, and shows that Coldplay and Dave Matthews are similar and Phish is notable different.

The results in Table 2 indicate the algorithm has approximately 60% accuracy, which is not that much better than random selection (50-50). Peak performance at 66% is seen with 8 features. As in the previous test condition, the high deviations confirm the variability within a given group. These results suggest the algorithm has the potential to classify bands from the same genre, but is not achieving optimal performance with the training set and parameters used in this analysis.

# of Features	**Accuracy (Mean)**	**Std. Dev.**
1	57%	7.0%
2	60%	9.0%
4	54%	15.1%
6	63%	14.9%
8	66%	11.6%
10	63%	10.3%
20	64%	10.1%
30	61%	8.7%
40	60%	5.9%
50	61%	6.5%
60	44%	13.1%

Table 2. For Test 2, mean and standard deviation results from 9 trials at the given values of the "features" parameter." The algorithm has an over 63% accuracy rate for classification of bands in the same genre.

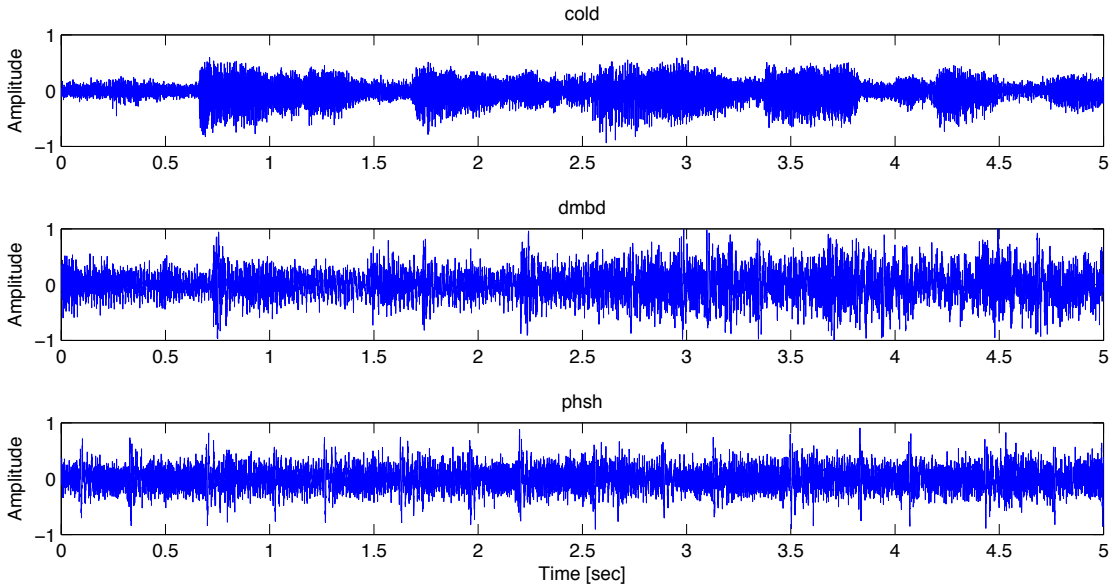

Figure 6. For Test 2, samples of the original audio show that Phish (bottom) and Dave Matthews (center) are very similar, and the track from Coldplay (top) is the only one with notable visual difference

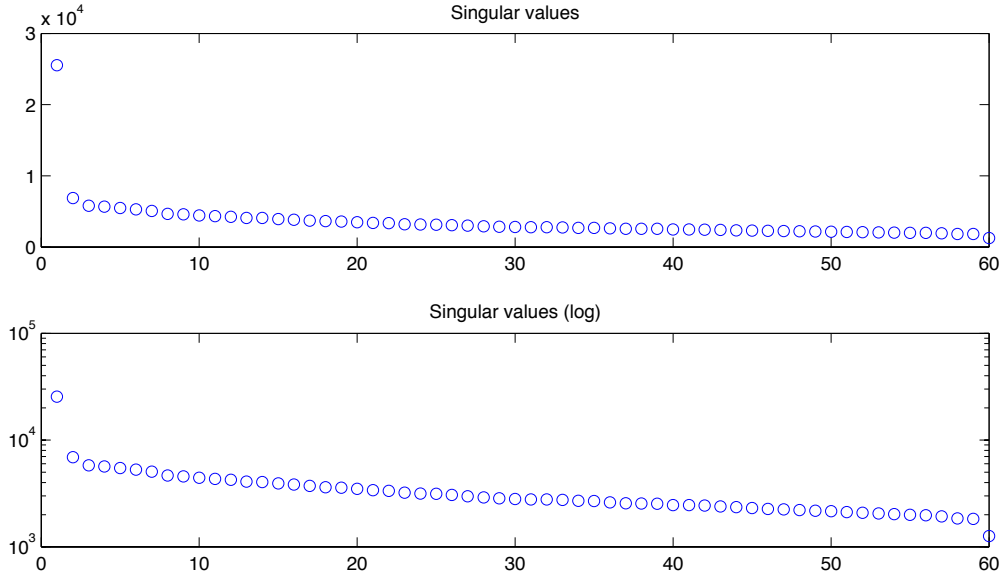

Figure 7. For Test 2, the first singular value is dominant, but there is a clear heavy-tail distribution.

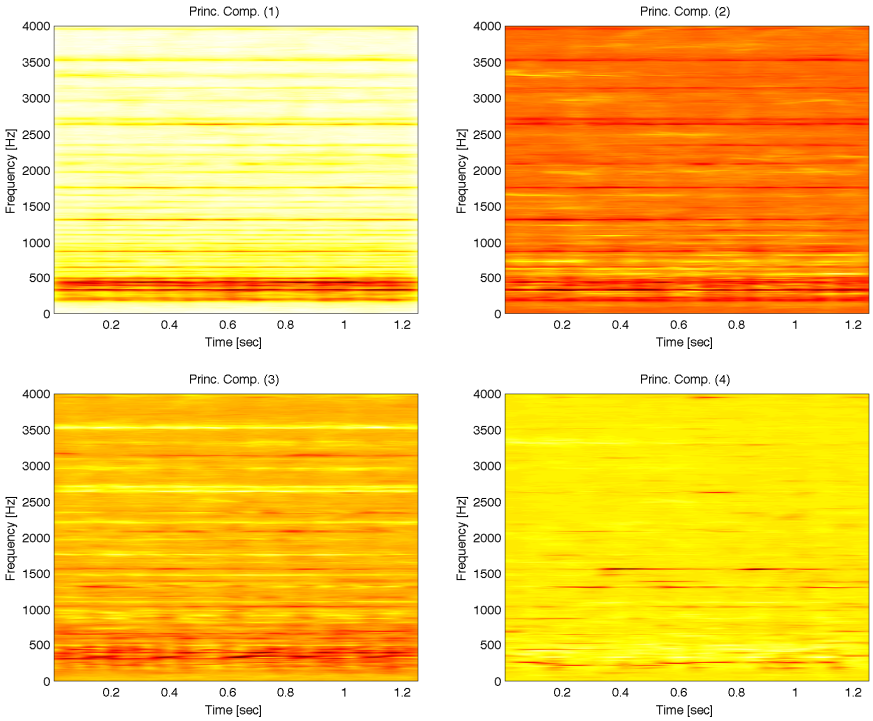

Figure 8. For Test 2, the first four principal components show distinct time-frequency signatures.

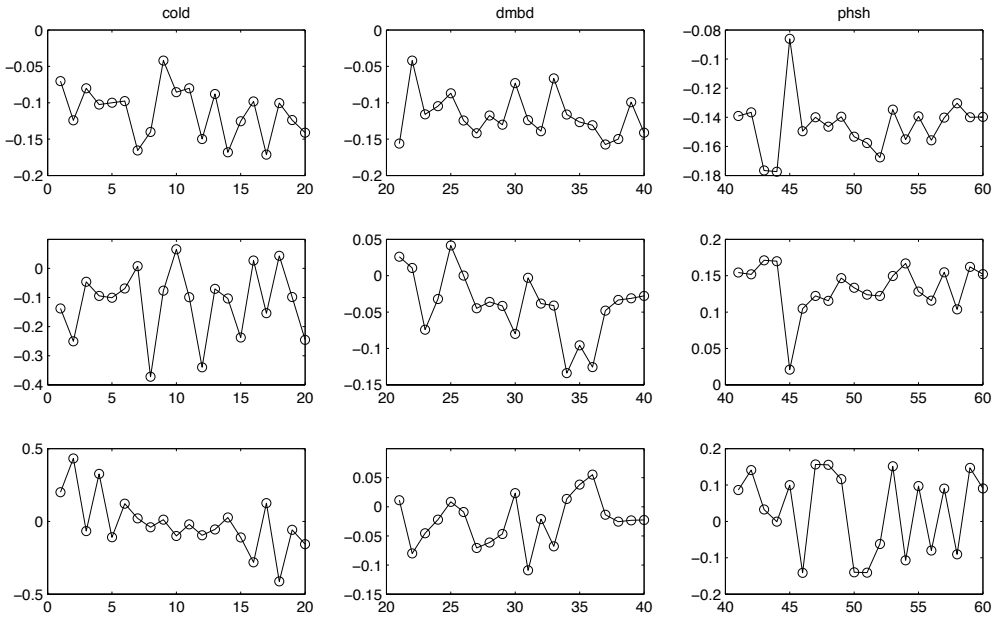

Figure 9. For Test 2, projection of the individual tracks onto the first three POD modes shows high intra-band variance and distinct signatures for each group: Coldplay (left), Dave Matthews (center), and Phish (right).

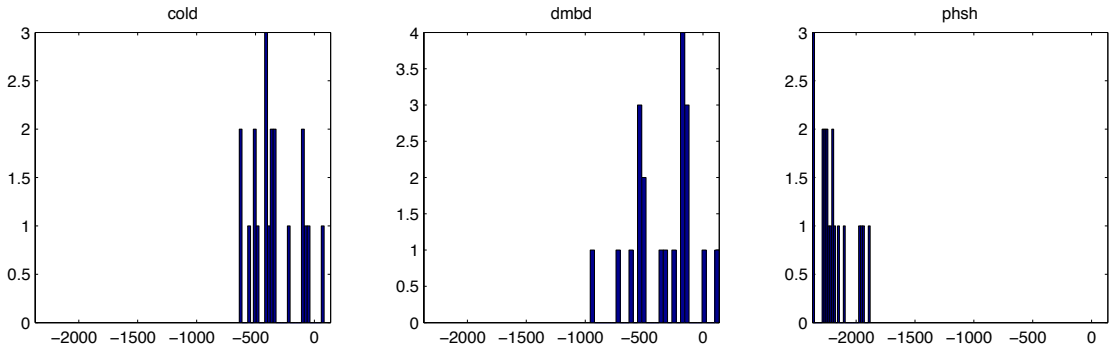

Figure 10. For Test 2, histograms of the projections shows high intra-band variance, that Coldplay (left) and Dave Matthews (center) are similar and Phish (left) is notable different.

4.3 Test 3

The objective of this test is to classify musical genres using a selection of bands within each genre. The three genres that tested are classical (Beethoven, Chopin, Rachmaninov, Mozart, and others), modern or alternative rock (Phish, Coldplay, Dave Mathews Band, and others) and techno (Fat Boy Slim, The Chemical Brothers, Moby, Yello, and others). As seen in Figure 13, the samples of the original audio show the three genres have the most distinct audio signatures in all the tests. As with the previous tests, the first singular value is still dominant but there is a clear heavy-tail distribution of the singular values (Figure 15). Spectrograms of the first four principal components (Figure 16) show distinct time-frequency signatures for each component. Projection of the individual tracks onto the first three POD modes (Figure 17) show high intra-genre variance as seen in the previous tests and distinct signatures for each genre. The histograms of the projections in Figure 18 confirms the high intra-genre variance, but also shows distinct signatures for each genre.

The results in Table 3 indicate the algorithm has approximately 65% accuracy, similar to test 2. Peak performance at 68% is seen with 10 features. As in the discussed previously, the deviations are high, but in this test they are consistently lower than the previous test conditions suggesting more accuracte results for this test. However, similar to Test 2 these results indicate the algorithm has the potential classify genres, but is not achieving optimal performance with the training set and parameters used in this analysis.

# of Features	Accuracy (Mean)	Std. Dev.
1	64%	6.5%
2	64%	7.2%
4	64%	9.8%
6	65%	9.1%
8	67%	6.9%
10	68%	7.1%
20	67%	7.0%
30	66%	8.3%
40	67%	9.2%
50	62%	9.2%
60	54%	12.8%

Table 3. For Test 3, mean and standard deviation results from 9 trials at the given values of the "features" parameter." The algorithm has an over 67% accuracy rate for classification of different genres.

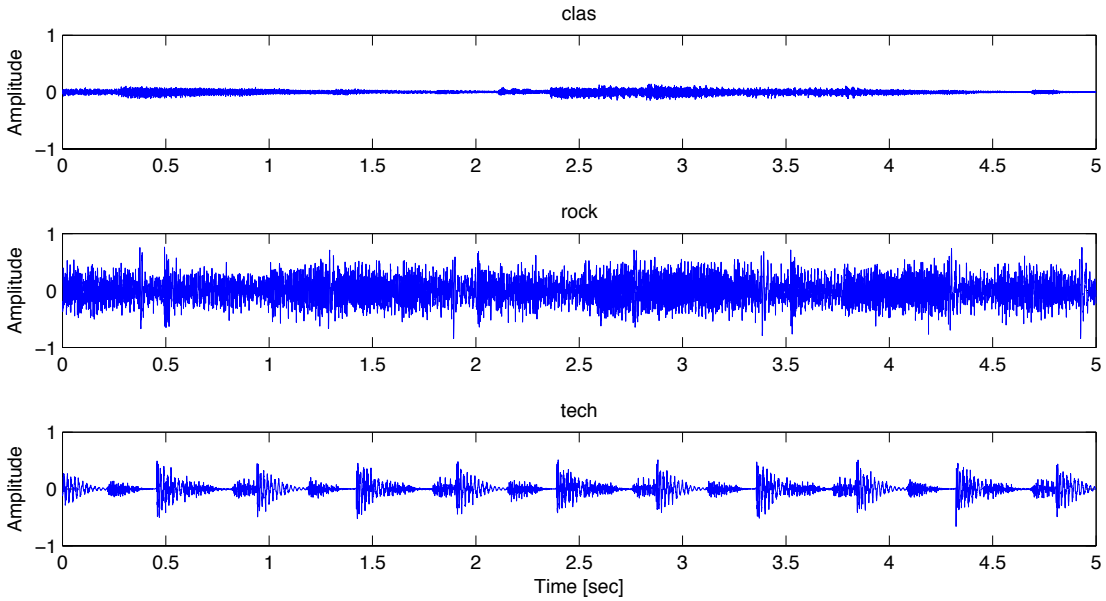

Figure 11. For Test 3, samples of the original audio show that the three genres classical (top), modern rock (center), and techno (bottom) have the most distinct audio signatures in all the tests.

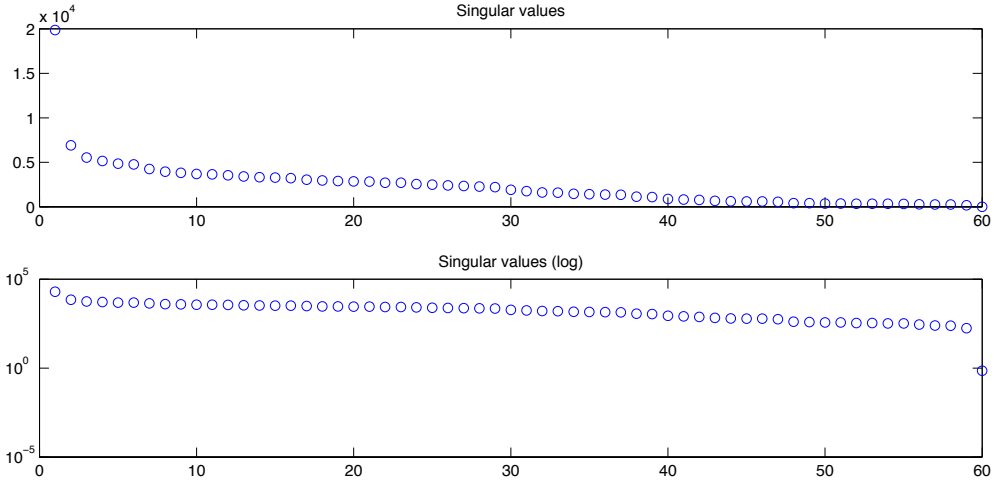

Figure 12. For Test 3, the first singular value is still dominant, and there is still a clear heavy-tail distribution.

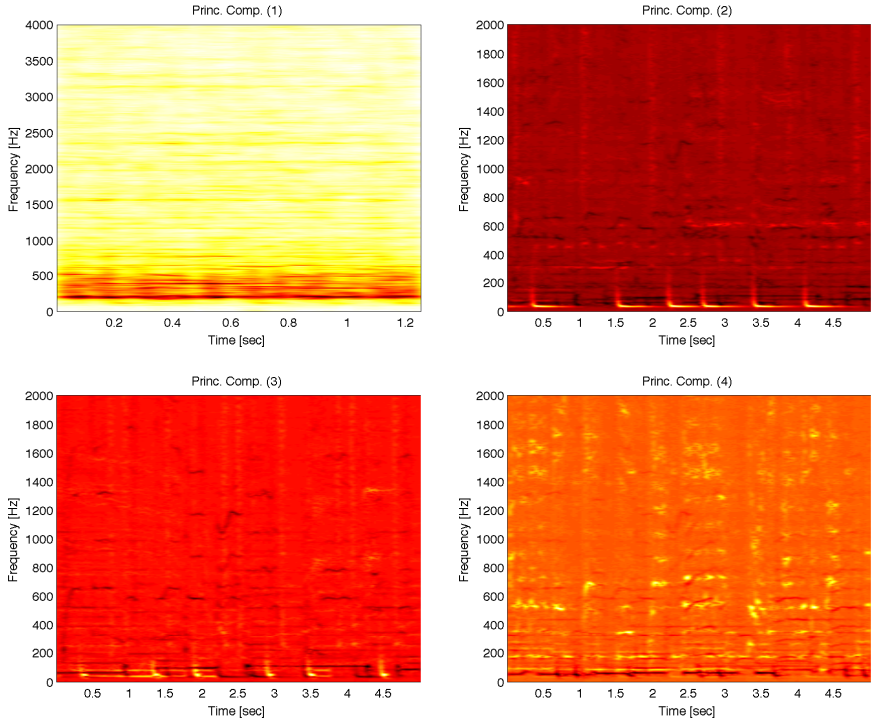

Figure 13. For Test 3, the first four principal components show distinct time-frequency signatures.

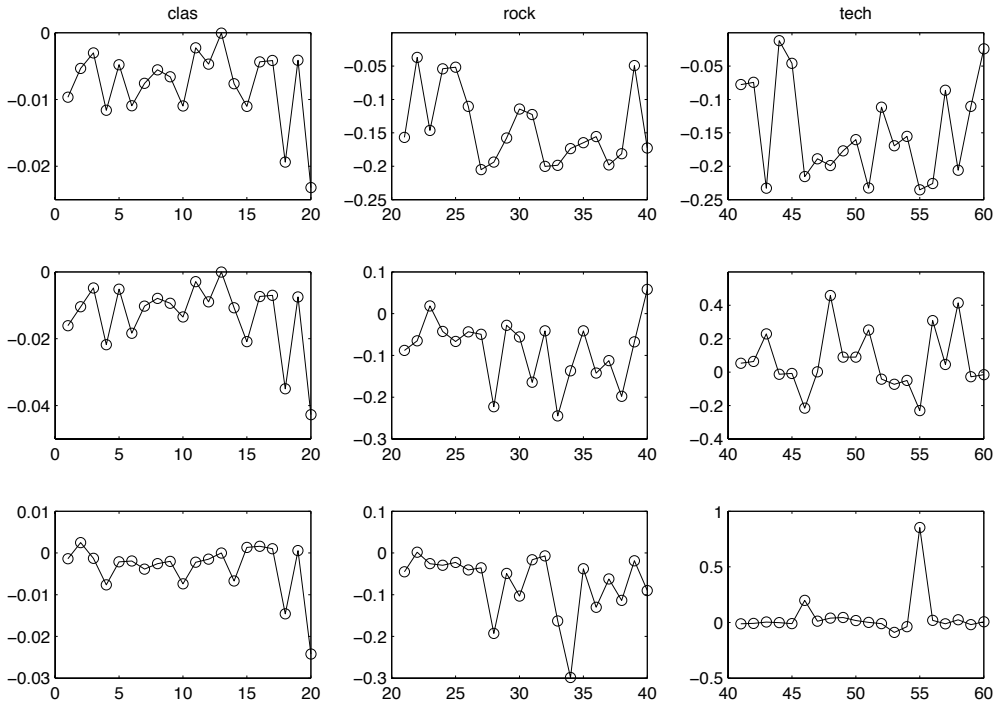

Figure 14. For Test 3, projection of the individual tracks onto the first three POD modes shows high intra-genre variance and distinct signatures for each genre: classical (left), modern rock (center), and techno (right).

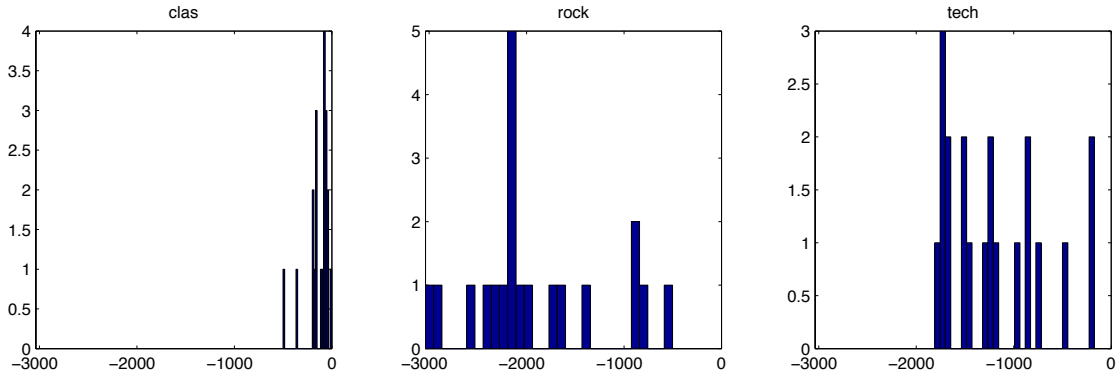

Figure 15. For Test 3, histograms of the projections shows high intra-genre variance and distinct signatures for each genre: classical (left), modern rock (center), and techno (right)

5 Summary and Conclusions

Ultimately, the objective in this project is to develop an algorithm to classify music genre and bands based on their time-frequency signatures. Using music from my iTunes collection, time-frequency analysis with the Gábor transform, singular value decomposition (SVD) , and linear discriminant analysis (LDA), this algorithm proves to be most effective with band classification from different genres and struggled with classification of different genres and bands within a single genre.

One limitation with this analysis is the number of tracks or data points used in the tests and training sets. Though the results above hint at the accuracy and performance of the algorithm, larger data sets would provide a more effective metric. However, there are memory management limitations to this algorithm and a balance is struck between resampling the incoming data to reduce size, maximizing the number of samples, and preventing memory overload in MATLAB.

An integral part of this project is the exploration of the methods, and so the number of time-frequency "features" (identified as modes from the SVD) necessary for optimal classification is extensively tested. One of the primary values of the SVD method is that it provides low rank approximations for a given dataset. This algorithm is run for 1, 2, 4, 6, 8, 10, 20, 30, 40, 50, and 60 "features" or modes. Since the training set contains 60 tracks then using 60 modes corresponds to a full rank. Interestingly, approximately 10 modes proved to be the most effective for all three test conditions. This may be related to some intrinsic structure of recorded music and further exploration of this phenomena is an interesting application of this algorithm.

Appendix A (MATLAB functions)

diag Used to extract the diagonals from a matrix.

double Used to convert 8-bit integer image data to double precision format.

eig Used to solve the eigenvalue problem; the function returns the eigenvalues and eigenvectors used for the LDA.

fftn The all important FFT function, which performs a discritized Fourier Transforms. This version of the function transforms n-dimensional data. The `fft` and `fft2` versions transform 1- and 2-dimensional data respectively.

fftshift The output of the `fft` algorithm is shifted (butterfly algorithm), so data in the frequency domain is shifted back using this function before plotting.

frame2im Used to covert a movie frame to an image.

length Used to get the length of vectors.

load Loads data from an external file.

ind2sub Used to convert a linear index (in this case the minimum value in the difference image) into corresponding 2D subscripts.

linspace Used to build a linear vector with $n + 1$ points for the spatial domain. The vector is then trimmed to n points due to the periodic boundaries.

max Used to find the maximum value in a given array.

mean Used to calculate the mean for each row of the location coordinate data.

num2str Used to convert a number to a string.

pcolor Used to plot the 3D spectrograms.

plot Used to plot the various parameters agains time or frequency.

repmat Used to replicate and tile an array for the means.

reshape Reshapes vector or matrix for given dimensions.

semilogy Used to make a plot with the y-axis on a logarithmic scale.

size Used to get the number of rows and columns for a given matrix.

strcat Used to concatenate a string.

subplot Used to produce plot arrays.

sum Used to calculate sums.

svd The all important SVD function used to obtain the U, S, and V matrices.

switch Conditional structure used to vary code execution based on a specified tag.

tic/toc Used to time the operations.

zeros Used to build vectors and matrices filled with "0".

Appendix B (code)

See project root directory for scripts.

Main

Main control script `main.m`.

Gabor

Subroutine (`gabor.m`) for the Gabor Transform.

Trainer

Subroutine (`trainer.m`) for building the training set and LDA calculations.

Importing Data

Subroutine (`loadTrack.m`) for importing track data.

Plotting Frequency Data

Subroutine (`plot_freq.m`) for plotting 2D frequency data.

Plotting Histogram of Results

Subroutine (`plot_histo.m`) for plotting LDA results.

Plotting LDA Results

Subroutine (`plot_lda.m`) for plotting LDA results.

Plotting Spectrograms

Subroutine (`plot_spectro.m`) for plotting spectrograms.

Appendix C (calculations)

None

Appendix D (data)

Here is the data obtained for the testing the "features" (modes) parameter used for SVD/LDA:

Test 1											
Features	**Trial 1**	**Trial 2**	**Trial 3**	**Trial 4**	**Trial 5**	**Trial 6**	**Trial 7**	**Trial 7**	**Trial 9**	**Mean**	**Std. Dev.**
1	83.3	73.3	83.3	73.3	70.0	66.7	83.3	70.0	86.7	**77**	7.5
2	73.3	83.3	73.3	63.3	73.3	70.0	43.3	56.7	60.0	**66**	11.8
4	83.3	90.0	56.7	83.3	80.0	73.3	73.3	76.7	90.0	**79**	10.3
6	90.0	90.0	66.7	80.0	90.0	73.3	70.0	83.3	93.3	**82**	9.9
8	90.0	90.0	76.7	70.0	90.0	66.7	66.7	86.7	90.0	**81**	10.6
10	93.3	90.0	76.7	73.3	83.3	73.3	80.0	86.7	86.7	**83**	7.2
20	93.3	73.3	43.3	76.7	86.7	60.0	80.0	86.7	86.7	**76**	15.8
30	83.3	76.7	80.0	53.3	76.7	56.7	86.7	83.3	80.0	**75**	11.9
40	76.7	90.0	86.7	63.3	53.3	60.0	76.7	70.0	86.7	**74**	13.0
50	73.3	40.0	73.3	56.7	73.3	83.3	63.3	53.3	93.3	**68**	16.2
60	66.7	60.0	50.0	46.7	76.7	66.7	50.0	66.7	40.0	**58**	12.0

Test 2											
Features	**Trial 1**	**Trial 2**	**Trial 3**	**Trial 4**	**Trial 5**	**Trial 6**	**Trial 7**	**Trial 7**	**Trial 9**	**Mean**	**Std. Dev.**
1	53.3	70.0	56.7	50.0	53.3	56.7	50.0	56.7	66.7	**57**	7.0
2	56.7	50.0	70.0	63.3	66.7	50.0	50.0	73.3	63.3	**60**	9.0
4	63.3	50.0	80.0	33.3	56.7	33.3	50.0	66.7	56.7	**54**	15.1
6	60.0	83.3	83.3	60.0	40.0	60.0	50.0	76.7	56.7	**63**	14.9
8	60.0	86.7	83.3	63.3	56.7	60.0	56.7	70.0	56.7	**66**	11.6
10	60.0	86.7	53.3	63.3	56.7	66.7	56.7	70.0	56.7	**63**	10.3
20	53.3	86.7	56.7	60.0	56.7	60.0	66.7	70.0	63.3	**64**	10.1
30	50.0	56.7	60.0	73.3	50.0	63.3	66.7	73.3	60.0	**61**	8.7
40	53.3	60.0	53.3	56.7	56.7	63.3	66.7	70.0	63.3	**60**	5.9
50	56.7	66.7	53.3	56.7	53.3	66.7	66.7	70.0	60.0	**61**	6.5
60	43.3	53.3	23.3	36.7	36.7	50.0	66.7	33.3	53.3	**44**	13.1

Test 3											
Features	**Trial 1**	**Trial 2**	**Trial 3**	**Trial 4**	**Trial 5**	**Trial 6**	**Trial 7**	**Trial 7**	**Trial 9**	**Mean**	**Std. Dev.**
1	70.0	63.3	63.3	56.7	66.7	60.0	66.7	56.7	76.7	**64**	6.5
2	70.0	66.7	70.0	73.3	56.7	60.0	66.7	56.7	53.3	**64**	7.2
4	70.0	63.3	53.3	70.0	66.7	60.0	63.3	46.7	80.0	**64**	9.8
6	70.0	63.3	46.7	76.7	66.7	60.0	63.3	63.3	76.7	**65**	9.1
8	70.0	63.3	63.3	76.7	66.7	60.0	66.7	56.7	76.7	**67**	6.9
10	70.0	63.3	60.0	76.7	66.7	60.0	70.0	63.3	80.0	**68**	7.1
20	70.0	60.0	63.3	73.3	66.7	56.7	66.7	66.7	80.0	**67**	7.0
30	73.3	66.7	66.7	76.7	56.7	56.7	56.7	76.7	63.3	**66**	8.3
40	76.7	73.3	70.0	76.7	63.3	56.7	66.7	73.3	50.0	**67**	9.2
50	53.3	50.0	73.3	73.3	63.3	53.3	70.0	70.0	53.3	**62**	9.7
60	46.7	46.7	33.3	66.7	60.0	50.0	70.0	46.7	70.0	**54**	12.8