Lab 2

Michael Eller

February 15, 2016

Contents

1 Introduction

In the first lab, we explored some of the many techniques used in audio industry to organize and search through music by content. We developed identifiers to attach to the music such as its loudness and frequency distribution. In this lab, we will be classifying tracks according to the identifiers developed in the first lab. Since many of the calculations were operating independently for each frame, parallelizing the process using MATLAB's `parfor` function, we are able to divide the processes unto multiple threads. Personally, this reduced my computation times by 50%, despite parallelization overhead.

Assignment

1. Modify your function that computes the **mfcc** to return the **mfcc** in decibels.
2. Construct a two-dimensional *spectrum histogram* (a matrix of counts), with 40 columns - one for each mfcc index, and 50 rows - one for each amplitude level, measured in dB, from -20 to 60 dB. You will normalize the histogram, such that the sum along each column (for each "note") is one.
3. Display, using the MATLAB function **imagesec**, the spectrum histogram for the 12 audio tracks supplied for the first lab.

2 Rhythm

Rhythm can be defined as the presence of repetitive patterns of movement or sound. In this lab, we will be using several techniques to detect the presence of rhythm in an audio track.

2.1 Spectrum Histogram

Extending from Lab1, we slightly modify our function that computes the **mfcc** coefficients in order to add a non-linearity. We will, however, only be using this nonlinear **mfcc** function in calculating the spectrum histograms.

Listing 1: mfcc.m

```
function [ mfccp ] = mfcc( fbank,Xn,~)
%mfcc Computes the Mel frequency coeffecients
%   The mel-spectrum (MFCC) coefficient of the n-th frame is defined for
%   p = 1,...,NB
narginchk(2, 3);
% Xn=freqDist(filename);
mfccp=(fbank*abs(Xn)).^2;
% Lets us still use non-log mfcc (if needed)
if(nargin==3)
    mfccp=10*log10(mfccp);
    return;
end
end
```

In order to better visualize the **mfcc** output, we create a *spectrum histogram* which maps each mfcc index and the corresponding amplitude levels to a color displaying how often each mfcc index occured at the respective volume. For example, in Figure 1, aside from the very low sounds, most of the pitches were at around the same volume. It is interesting to note in Figure 9 in Appendix A.1, there is a high concentration at the lower left corner. This makes sense though.The song has practically no bass to it, and the Spectrum Histogram reflects that. There should also exist a high similarity along the main diagonal, as a frame is completely similar to itself.

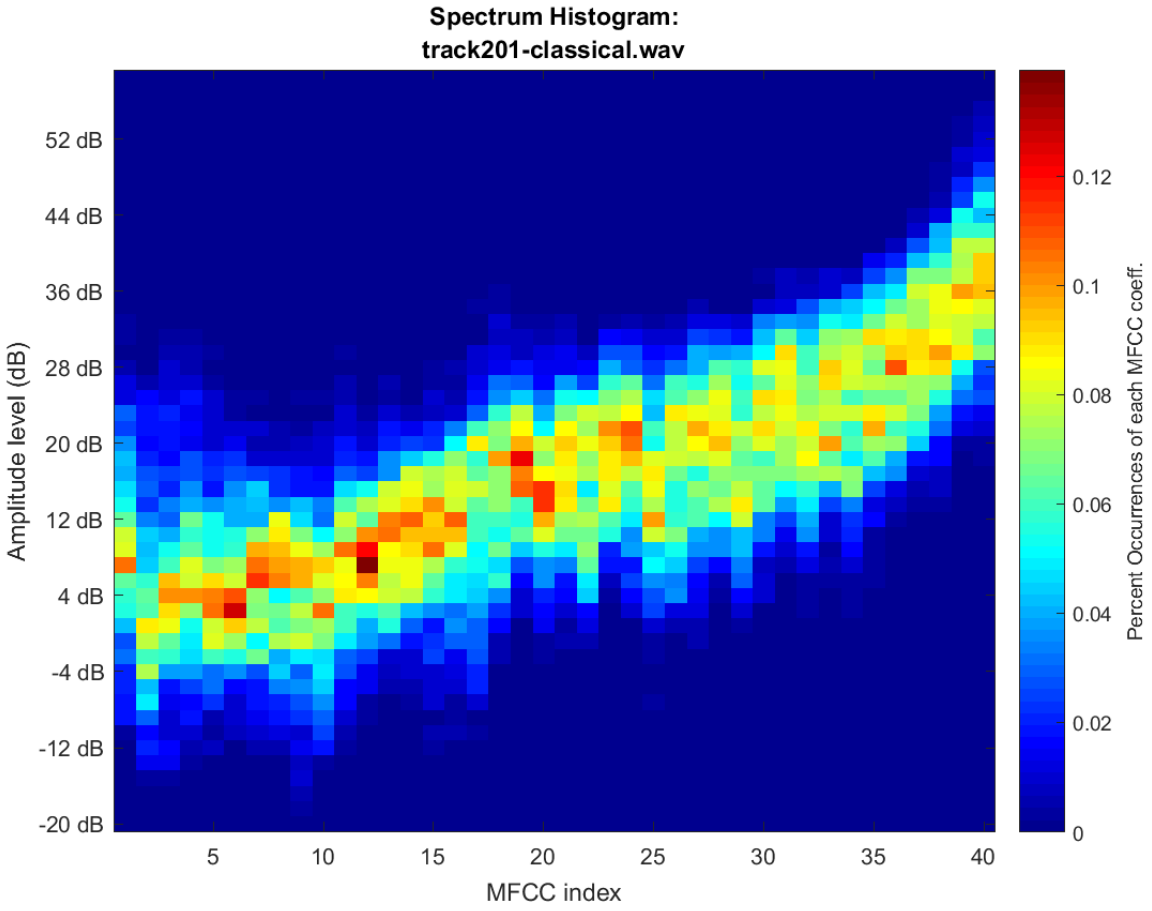

Figure 1: Spectrum Histogram 201

2.2 Similarity Matrix

Assignment

4. Implement the computation of the similarity matrix, and display the similarity matrices (color-coded) for the 12 audio tracks supplied for the first lab.

The accompanying code and figure for the similarity matrix for "track201-classical.wav" can be seen below. The remaining figures can be found in Appendix A.2.

Listing 2: simMatrix.m

```
function [ sim ] = simMatrix(mfccp,filename,~)
%simMatrix displays the similarity matrix for the mel coefficients
%   The similarity matrix is defined as follows:
%   S(i,j) = sum_{k=1}^{nbanks}{
%   (mfcc(k,i)*mfcc(k,j))/(norm(mfcc(:,i)) * norm(mfcc(:,j)))
% fbank=melBank();
% mfccp=mfcc(fbank,filename);
[a,b]=size(mfccp);
% Preallocate matrix
sim=zeros(b,b);
```

```
normI=sqrt(sum(abs(mfccp).^2,1));
parfor i=1:b %frame i
    for j=1:b %frame j
        for k=1:a %fbank k
            sim(i,j)=sim(i,j)+...
                (mfccp(k,i)*mfccp(k,j)/(normI(j).*normI(i)));
        end
    end
end

if(nargin == 3)
    h=figure;
    imagesc(sim);
    xlabel('Frame Number');
    ylabel('Frame Number');
    title({'Similarity Matrix:'; filename});
    colorbar;
    colormap 'jet';
    saveas(gca,['SimMatrix' filename(6:end-4) '.png']);
    close(h);
end
end
```

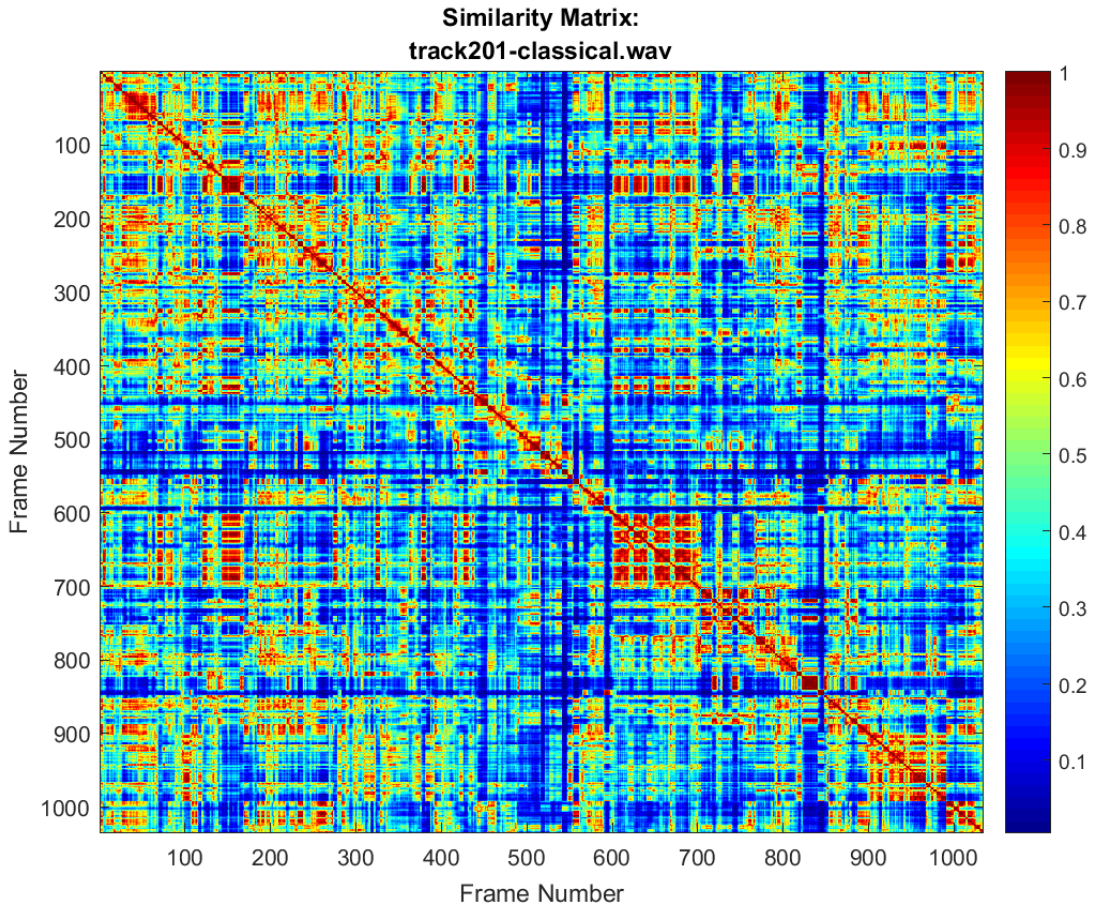

Figure 2: Similarity Matrix 201

The first stage involves computing a similarity matrix defined as follows

$$S(i,j) = \frac{\langle \text{mfcc}(:,i), \text{mfcc}(:,j) \rangle}{||\text{mfcc}(:,i)||\,||\text{mfcc}(:,j)||} = \sum_{k=1}^{\text{nbanks}} \frac{\text{mfcc}(k,i)\ \text{mfcc}(k,j)}{||\text{mfcc}(:,i)||\,||\text{mfcc}(:,j)||} \tag{1}$$

Currently the similarity matrices are not quite correct. While they do possess all the correct features in that they are symmetrical, their outputs do not make sense. According to my function, every frame appears extremely similar. With an ideal function, one would be able to see *hotspots* on the map, corresponding to longer notes, with the size of each hotspot being inversely proportional to the speed of the rhythmic progression. Using Figure 2 as an example, it would imply that near the beginning of the piece there is a general rhythmic pattern. But as the piece progresses past the 300^{th} frame, the similarity drastically decreases. The musical equivalent of this would be the piece has sped up insanely fast and now there is a different pitch almost every $\frac{1}{f_s \times 255}$ (0.0231) seconds. This would imply that there were up to 43 new pitches every second.

2.3 A first Estimate of the Rhythm

Once we note that the entries in our similarity matrix are always at a distance of l frames, we are able to better quantify the rhythm by creating a rhythm index, defined by Equation 2.

$$B(l) = \frac{1}{N_f - l} \sum_{n=1}^{N_f - l} S(n, n+l), \quad l = 0, \ldots N_f - 1 \tag{2}$$

Assignment

5. Implement the computation of the rhythm index $B(l)$ defined in Equation 2. Plot the vector B as a function of the lag $l = 0, N_f - 1$ for the 12 tracks. Comment on the presence, or absence, of a strong rhythmic pattern.
   Hint: to debug your function, you can use track-396 which should display strong rhythmic structure.

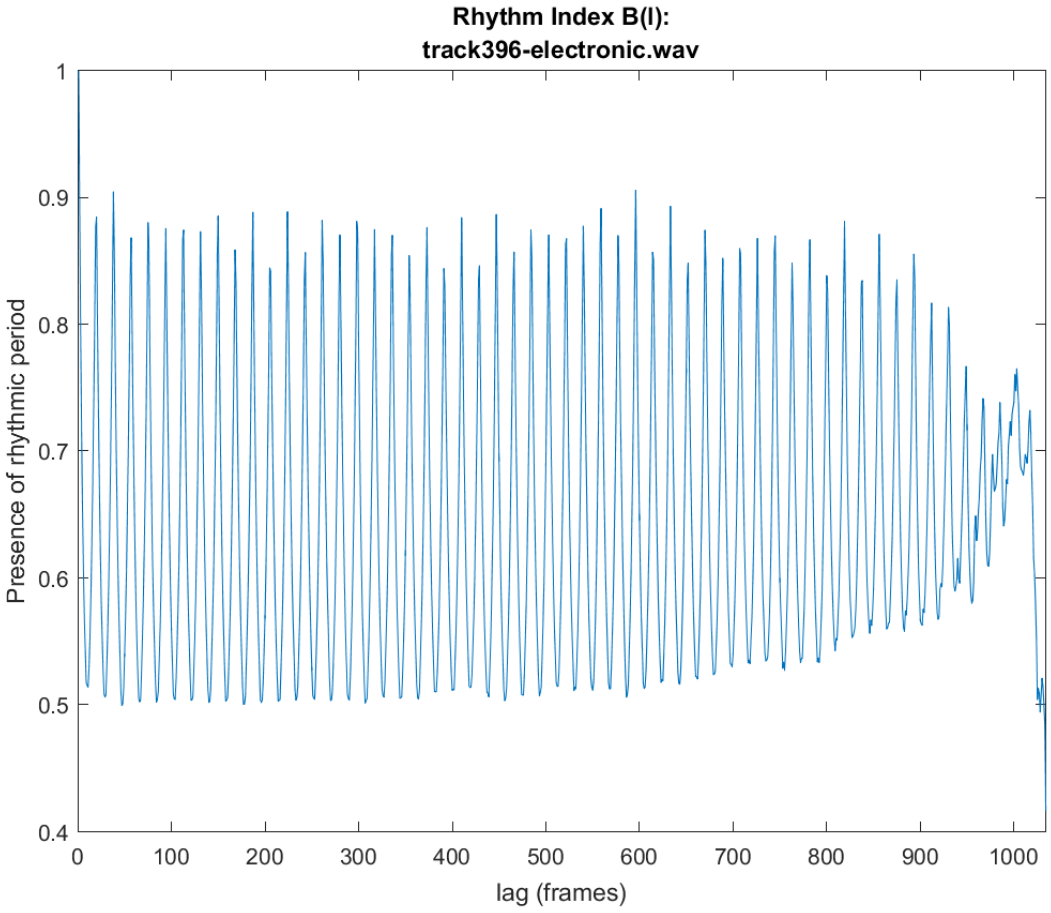

Figure 3: Rhythm Index 396

As can be seen in Figure 3, There is a very periodic pattern throughout the song. The periodicity of the graph in Figure 3 may hint at the underlying subdivision of the beat. This makes sense for track 396, as it has a very repetitive thumping bass line throughout the song. The spikes on the left side of the graph are more akin to faster notes while slower notes are represented further right.

2.4 A Better Estimate of the Rhythm

Rather than compounding the similarity between frames that are at a fixed lag apart in time, we can try to detect more interesting patterns in the similarity matrix. In addition to knowing the similarity between two frames, we would also like to know if this similarity is repeated later in the segment, at time j+l. The autocorrelation of frames can be defined as:

$$AR(l) = \frac{1}{N_f(N_f - l)} \sum_{i=1}^{N_f} \sum_{j=1}^{N_f - l} S(i,j)S(i,j+l), \quad l = 0, \ldots, N_f - 1 \tag{3}$$

Assignment

6. Implement the computation of the rhythm index $AR(l)$ defined in Equation 3. Plot the vector AR as a function of the lag $l = 0, N_f - 1$ for the 12 tracks. Comment on the presence, or absence, of a strong rhythmic pattern.

Equation 3 can be represented in MATLAB with Listing 3 below.

Listing 3: AutoC.m

```
function [ AR ] = autoC( filename,sim,~ )
%autoC Computes the autocorrelation of a song
%   In general, if two frames i and j are similar, we can find out
%   if they are repeated later in the segment, at time j+l

info = audioinfo(filename);
% sim = simMatrix( filename);
[len,~]=size(sim);
AR=zeros(1,len);

parfor l=0:len-1
    for i=1:len
        for j=1:len-l
            AR(l+1)=AR(l+1)+(sim(i,j)*sim(i,j+l));
        end
    end
    AR(l+1)=AR(l+1)*(1/(len*(len-l)));
end

if(nargin==3)
    h=figure;
    xAxis=linspace(0,len/info.SampleRate,len);
    plot(xAxis,AR);
    xlim([0,len/info.SampleRate]);
    title({'Autocorrelation AR(l):'; filename});
    xlabel('Lag (secs)');
    ylabel('Autocorrelation');
    saveas(gca,['AutoC' filename(6:end-4) '.png']);
    close(h);
end

end
```

The plot for the autocorrelation of track396-electronic.wav can be seen in Figure 4 below. The plot of the autocorrelation is strikingly similar to the plots for the rhythm index, B(l). At this point in analysis, the better choice would be whichever method is faster. The remaining figures can be found in Appendix A.4.

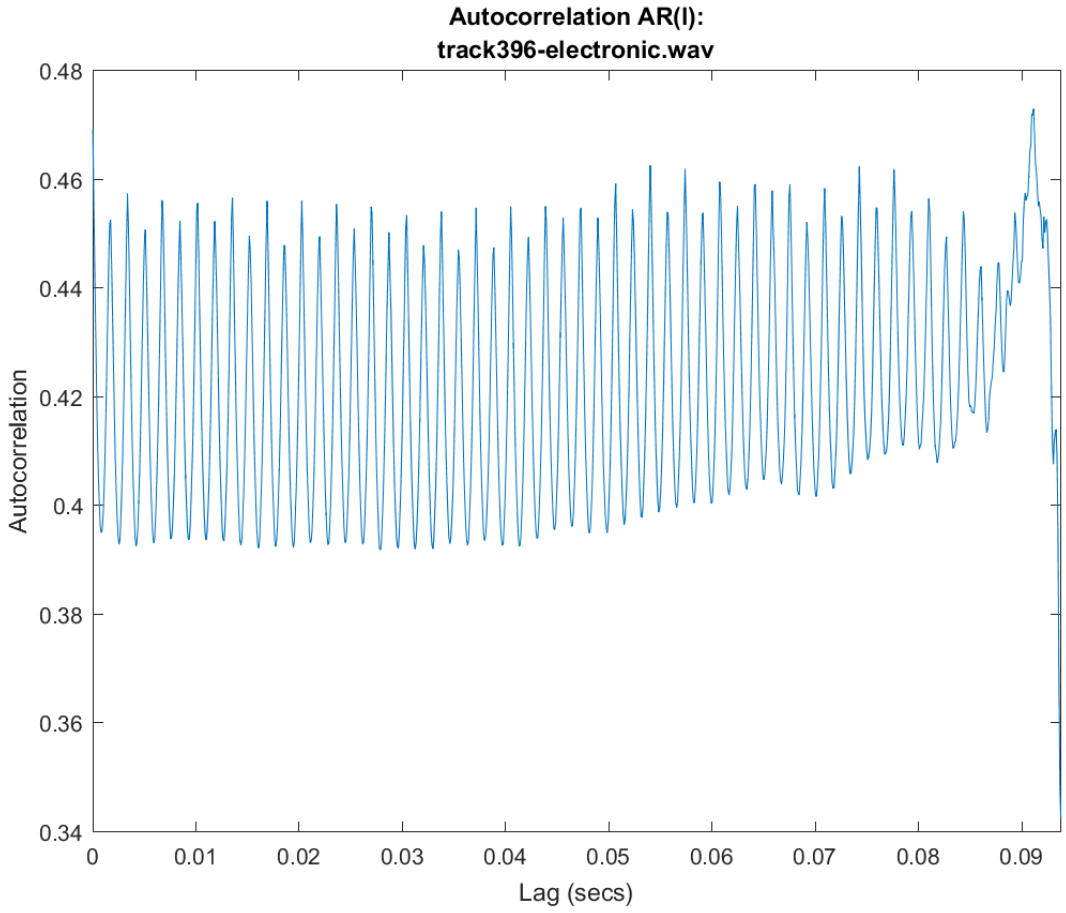

Figure 4: Autocorrelation 396

2.5 Rhythmic Variations over Time

Here we are interested in studying the dynamic changes in the rhythm. For this purpose, we consider short times windows formed by 20 frames (approximately 1 second) over which we compute a vector AR of size 20.

$$AR(l,m) = \frac{1}{20(20-l)} \sum_{i=1}^{20} \sum_{j=1}^{20-l} S(i+m*20, j+m*20) S(i+m*20, j+m*20+l) \quad (4)$$
$$\text{for } l = 0, \ldots, 19, \quad \text{and } m = 0, \ldots, N_f/20 - 1$$

Assignment

7. Implement the computation of the rhythm index $AR(l,m)$ defined in Equation 4. Plot the image AR in false colors as a function of the lag $l = 0, \ldots, 19$ (y-axis) and the time window index m (x-axis) for the 12 tracks. Comment on the rhythm patterns for the different tracks.

Equation 4 can be represented in MATLAB with Listing 4 below.

Listing 4: rhythmVar.m

```
function [ ARm ] = rhythmVar( filename,sim,~ )
%autoC Computes the autocorrelation of a song
%   In general, if two frames i and j are similar, we can find out
%   if they are repeated later in the segment, at time j+l
%   For the sake of this lab, we will be processing 20 frames (~1 second)
% sim = simMatrix( filename);
[len,~]=size(sim);
ARm=zeros(20,floor(len/20));

for m=0:floor(len/20-1)
    Window = sim(20*m+1:20*(m+1),20*m+1:20*(m+1));
    for l=0:19
        temp=Window.*circshift(Window,[0,-l]);
        ARm(l+1,m+1) = sum(sum(temp(1:20-l,:)))/((20-l));
    end
end


ARm=ARm/20;

if(nargin==3)
    h=figure;
    imagesc(ARm);
    ax=gca;
    title({'Autocorrelation AR(l,m):'; filename});
    xlabel('Time (secs)');
    ylabel('Lag (secs)');
    colormap 'jet';
    colorbar;
    saveas(gca,['RhythmVar' filename(6:end-4) '.png']);
    close(h);
end
end
```

Figure 5 shows an example of a song segment's rhythmic variation. My implementation, while currently incorrect, would provide useful information if it were right. Figure 5 shows an extremely quick song. All of the successive frames barely lag behind one another, meaning that they are extremely similar. Any rhythm would have to be occurring one after another. The remaining figures can be found in Appendix A.5.

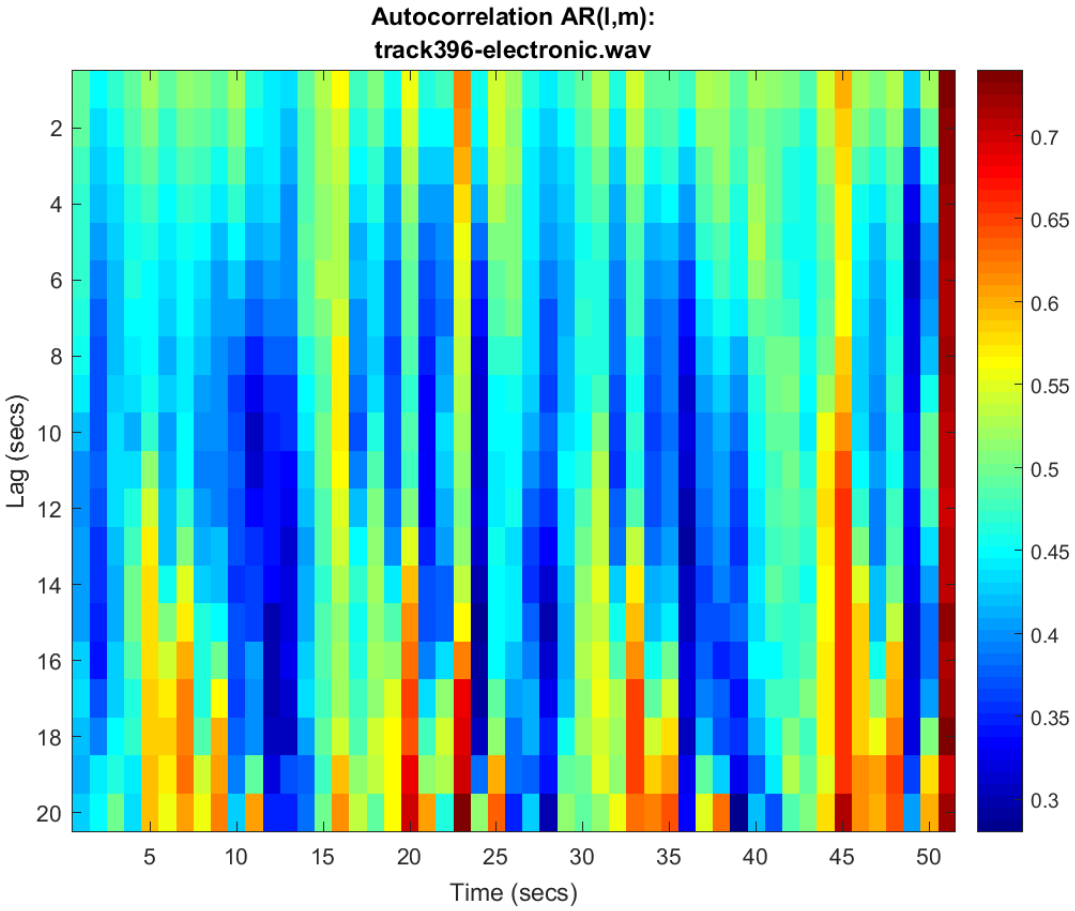

Figure 5: Rhythmic Variation 396

3 Tonality and Chroma

3.1 The Equal-Tempered Scale

Now that we have adequately explored the rhythmic aspect of a musical piece, it is time to look at the tonal portion of our music.

3.2 Chroma

The chroma is associated with the relative position of a note inside an octave. It is a relative measurement that is independent of the absolute pitch.

We define a *Pitch Class Profile* associated with the note, or chroma $c = 0, \ldots, 11$ as the weighted sum of all the peak frequencies that are mapped to the note c, irrespective of the octave they fall in.

$$PCP(c) = \sum_k w(k,c)|X_n(k)|^2 \qquad \text{where } k \text{ is such that } c = \text{round}(12\log_2(f_k/f_0)) \mod 12 \tag{5}$$

and the weight $w(k,c)$ is given by

$$w(k,c) = \begin{cases} \cos^2(\pi r/2) & \text{if } -1 < r = 12\log_2(f_k/f_0) - sm < 1, \\ & \text{where } c = sm \mod (12), \text{ and } sm = \text{round}(12\log_2(f_k/f_0)), \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

We want our definition of chroma to be independent of the loudness of the music. We therefore need to normalize each PCP by the total PCP computed over all the semitones. The normalized pitch class profile (NPCP) is defined by

$$\text{NPCP}(c) = \frac{PCP(c)}{\sum_{q=1}^{11} PCP(q)} \tag{7}$$

Assignment

8. Implement the computation of the Normalized Pitch Class Profile, defined by Equation 7. You will compute a vector of 12 entries for each frame n.
9. Evaluate and plot the NPCP for the 12 audio tracks supplied for the first lab.

With the NPCP function in Listing 5 and the resulting Figure 6, we are able to better examine the chromatic makeup of the song. For instance, it can be loosely inferred that track201-classical.wav is in the key of E based on the spectrogram. The most common not is E, until around frame 720. It then modulates to a new tonal center, perhaps D.

Listing 5: normPCP.m

```
function [PCP] = normPCP( filename,x,~)
%rhythmIndex Identifies rhythmic periods
%   B(l) is the sum of all the entries on the lth upper diagonal.
%   The index l associated with the largest value of B(l) corresponds to
%   the presence of a rhythmic period of l*K/f_s seconds
info=audioinfo(filename);
% x=extractSound(filename);
[s,~,~]=spectrogram(x,kaiser(512),256,512,info.SampleRate,'yaxis');
s=abs(s);
[a,b]=size(s);

% pks=zeros(a,b);
locs=zeros(a,b);
peaks=zeros(a,b);
for i=1:b
    len=length(findpeaks(s(:,i)));
    [peaks(1:len,i),locs(1:len,i)]=findpeaks(s(:,i));
end
freqVals=(5512/257)*locs;
f0=27.5;
sm=round(12*log2(freqVals/f0));
r=12*log2(freqVals/f0)-sm;
c=mod(sm,12);
[~,B]=size(c); % 257,1032

w=(cos(pi*r/2)).^2;
% Make pretty with matrix
w(isnan(w))=0;

% k is the number of peaks
% c is the chroma (A,A#,B,etc.)
% n is the frame number
```

```
% Weighting function, w, needs to be k*n*c large

PCP=zeros(12,B);


% Pick out semitones
% Loop through each value of c (1,2,...,12)
% Find the corresponding rows to extract from w where semitone is
% (1,2,...,12) multiply by Xn at that row at the corresponding row



peaks=peaks.^2;

for i=1:B %nFrames
    for j=0:11
        a=find(c(:,i)==j);
        PCP(12-j,i)=sum(w(a,i).*peaks(a,i));
    end
end
PCP=bsxfun(@rdivide,PCP,sum(PCP));



if(nargin==3)
    h=figure;
    imagesc(PCP);
    ax=gca;
    ax.YTickLabel=fliplr({'A ','A#','B ','C ','C#','D ','D#','E ',...
        'F ','F#','G ','G#'});
    ax.YTick=linspace(1,12,12);
    colormap 'jet';
    title({'Chroma :'; filename});
    xlabel('frames');
    ylabel('Note');
    colorbar;
    saveas(gca,['NPCP' filename(6:end-4) '.png']);
    close(h);
end
end
```

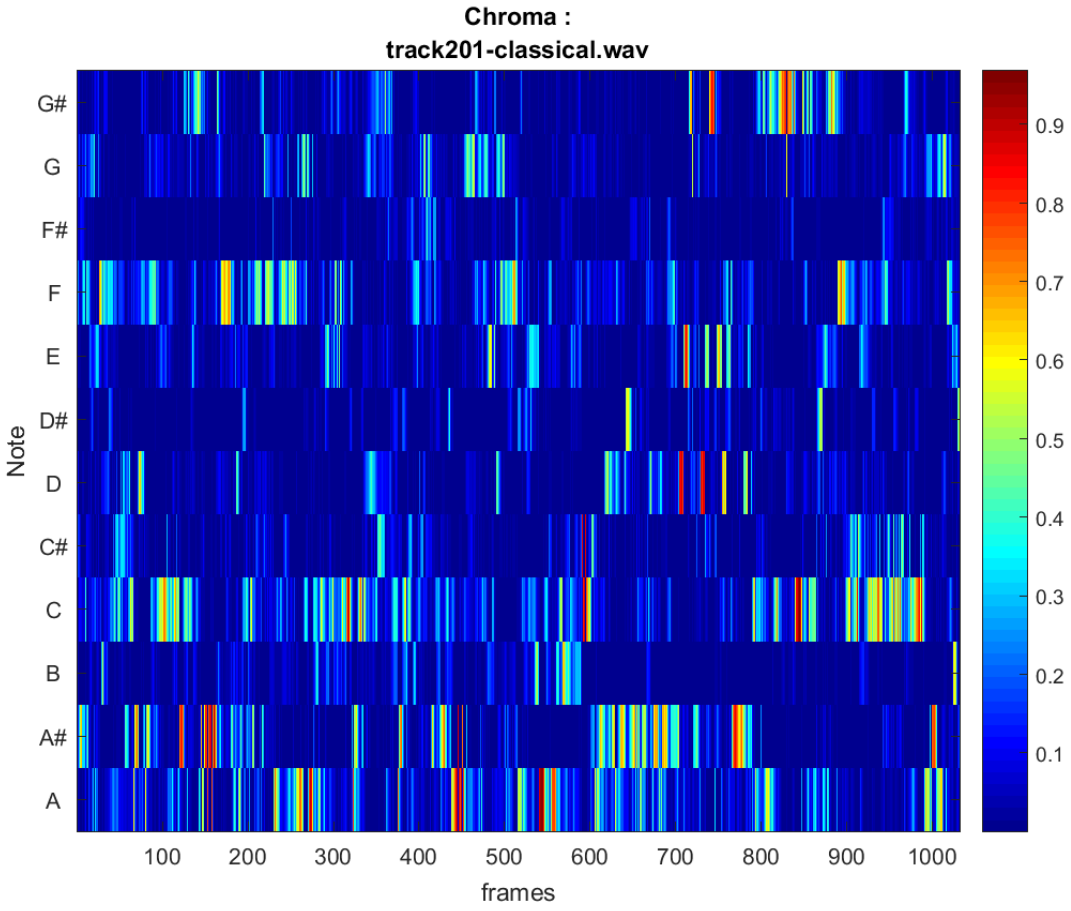

Figure 6: Pitch Class Profile 201

A Figures

A.1 Spectrum Histogram

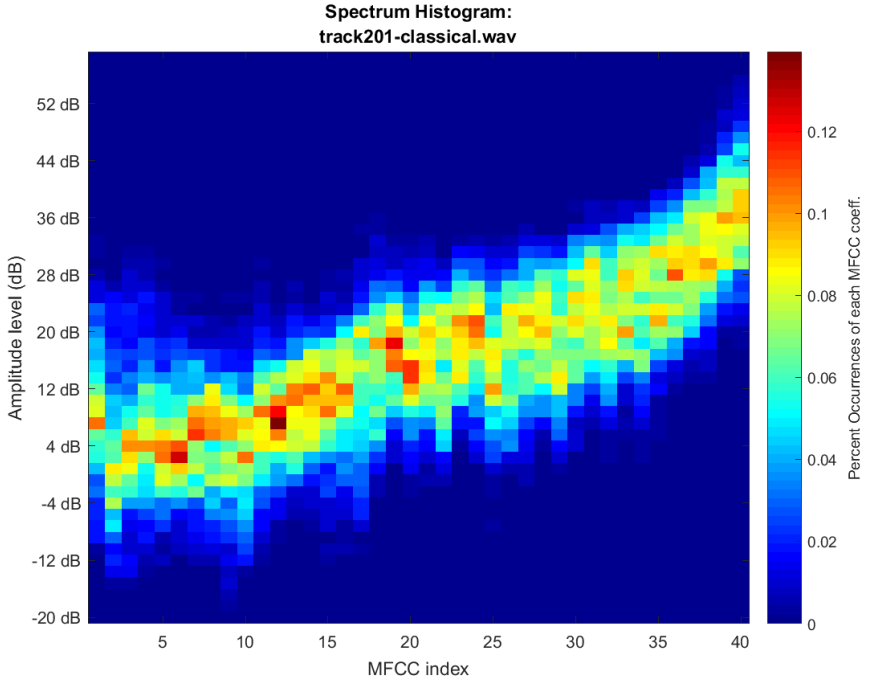

Figure 7: Spectrum Histogram 201

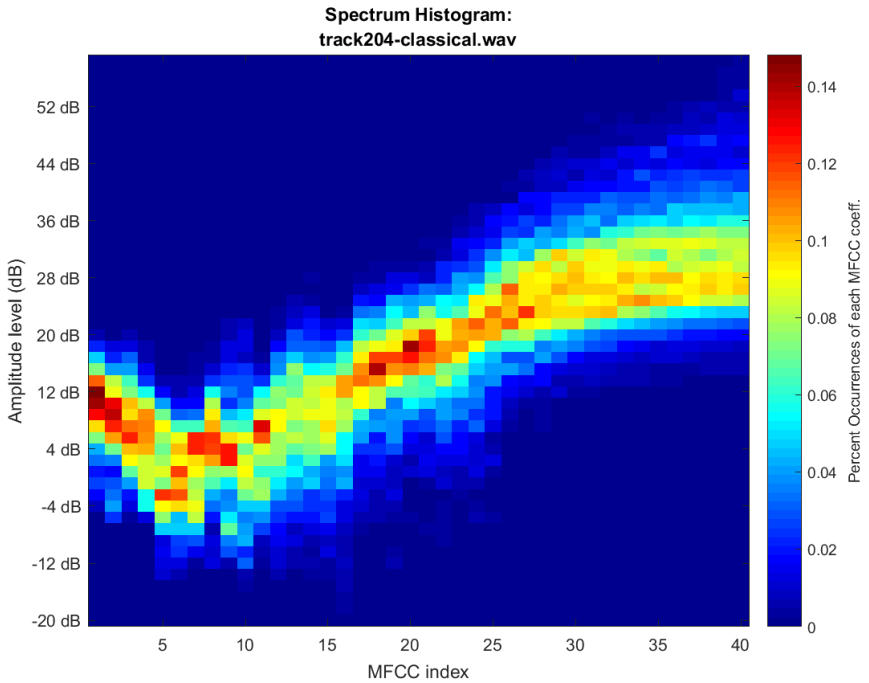

Figure 8: Spectrum Histogram 204

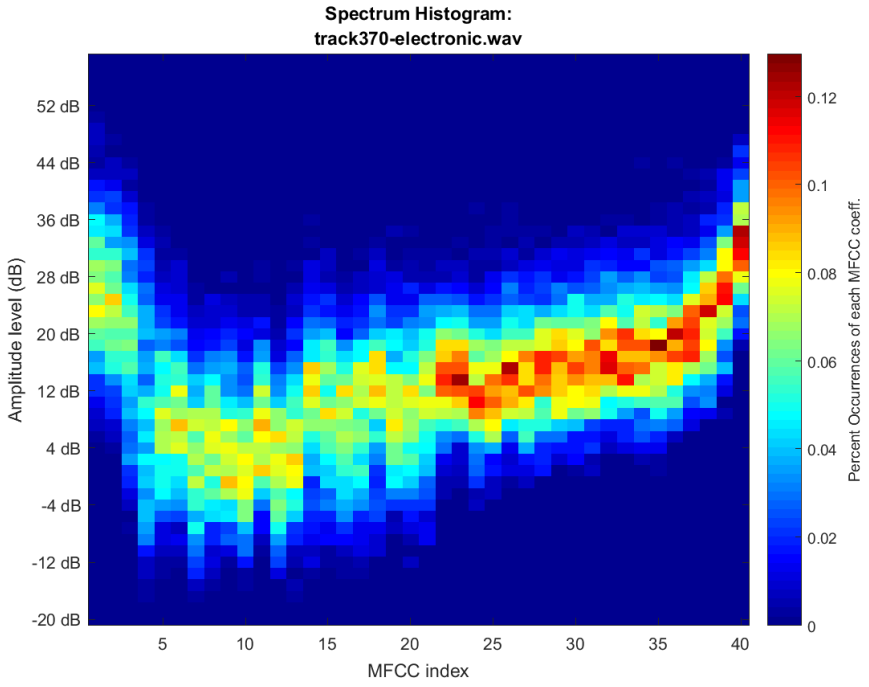

Figure 9: Spectrum Histogram 370

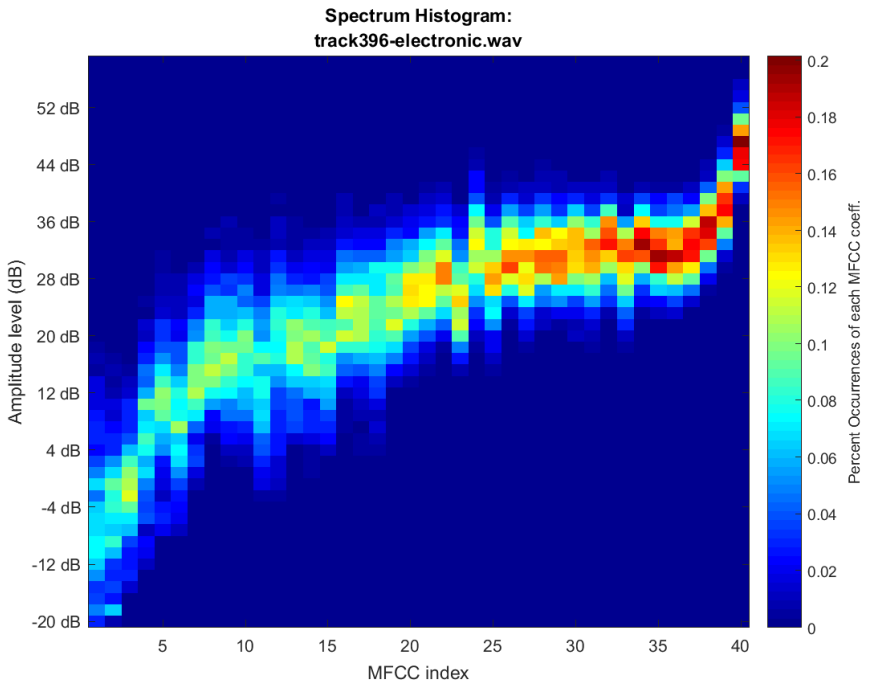

Figure 10: Spectrum Histogram 396

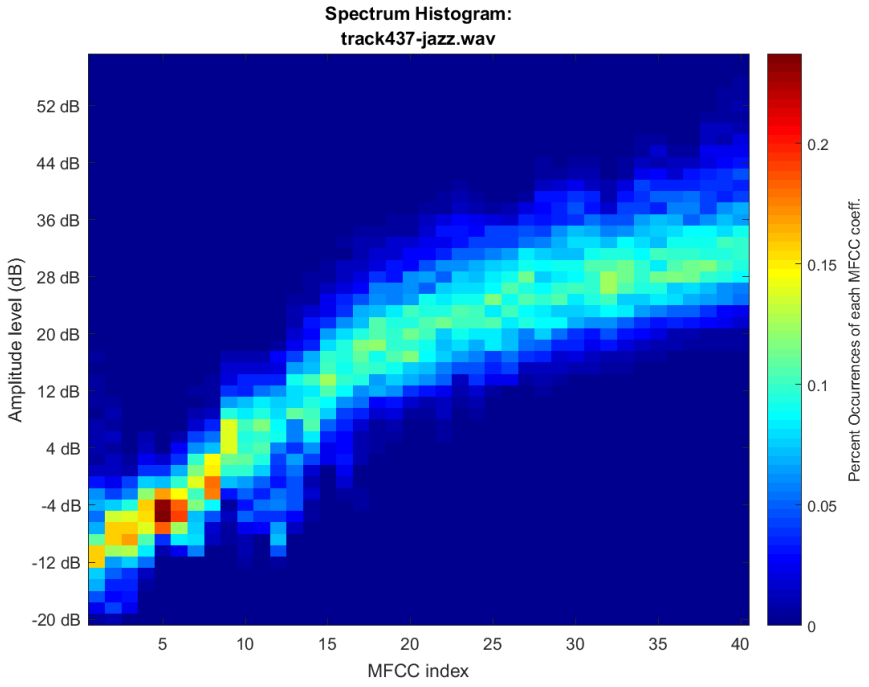

Figure 11: Spectrum Histogram 437

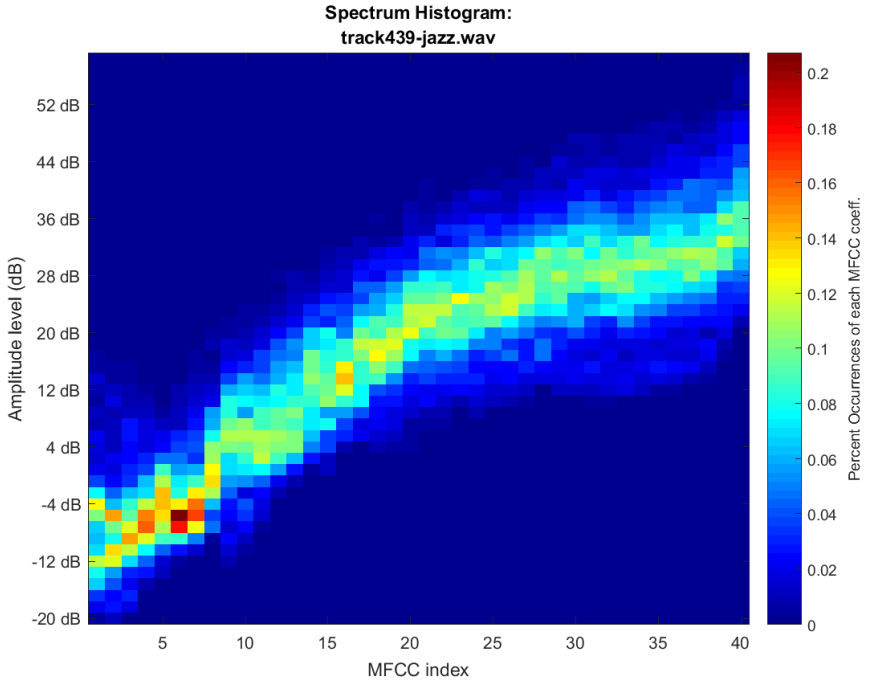

Figure 12: Spectrum Histogram 439

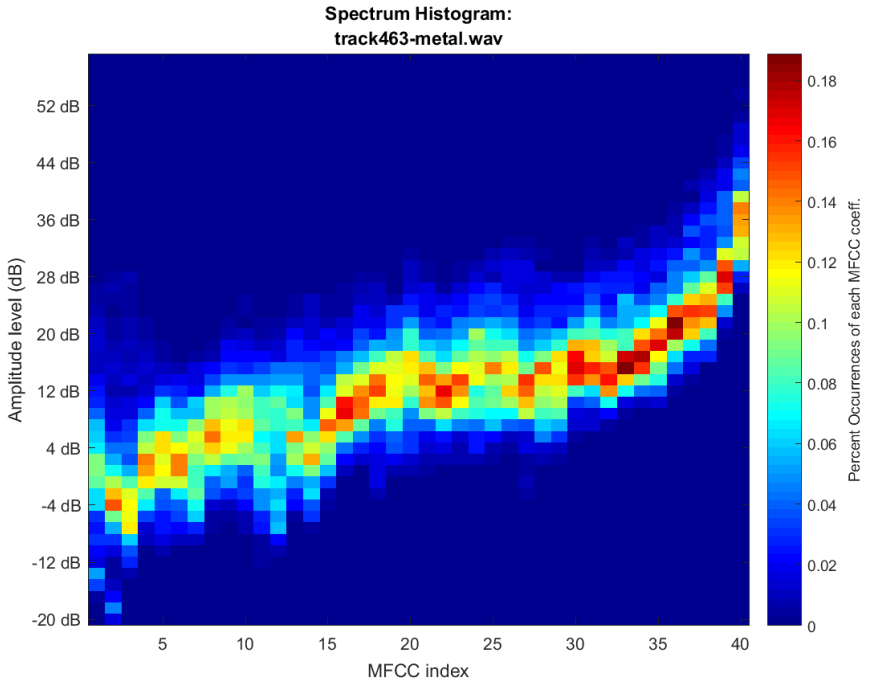

Figure 13: Spectrum Histogram 463

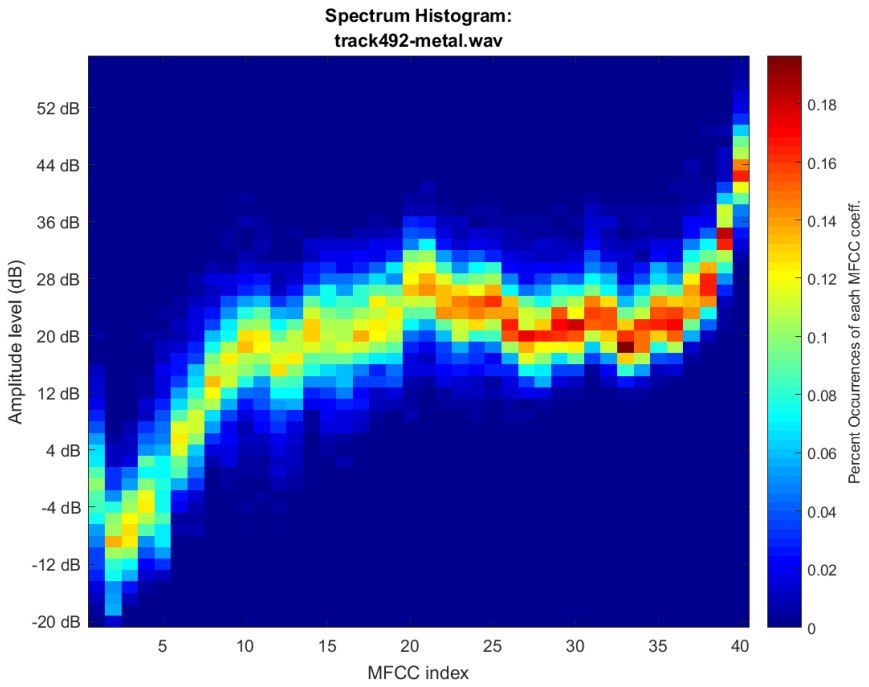

Figure 14: Spectrum Histogram 492

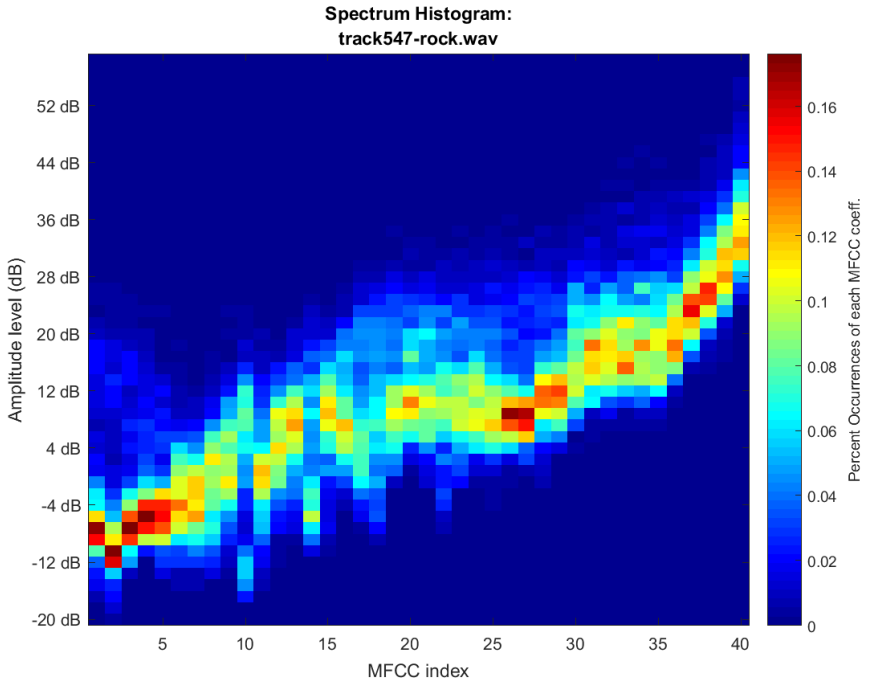

Figure 15: Spectrum Histogram 547

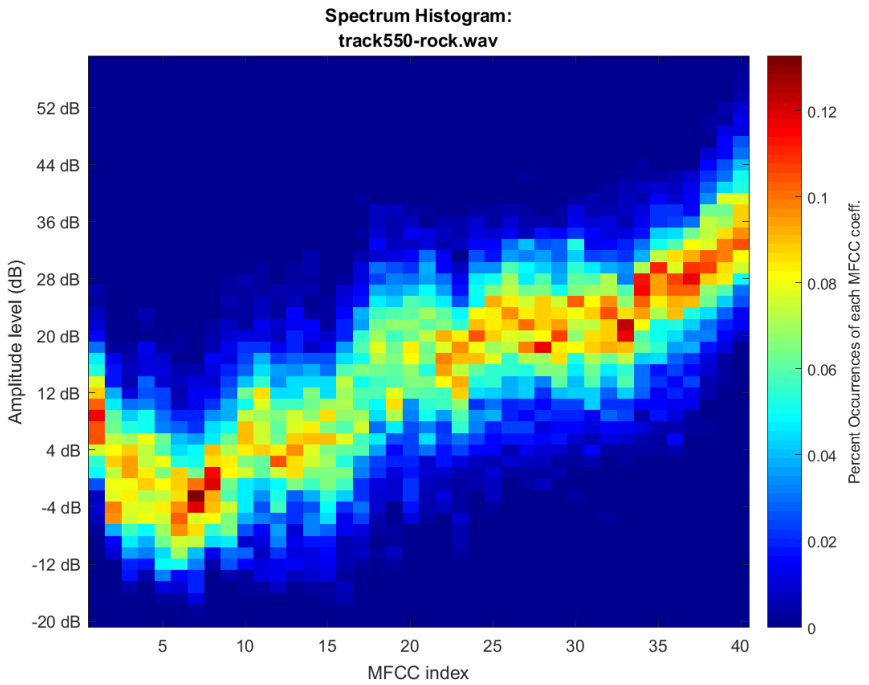

Figure 16: Spectrum Histogram 550

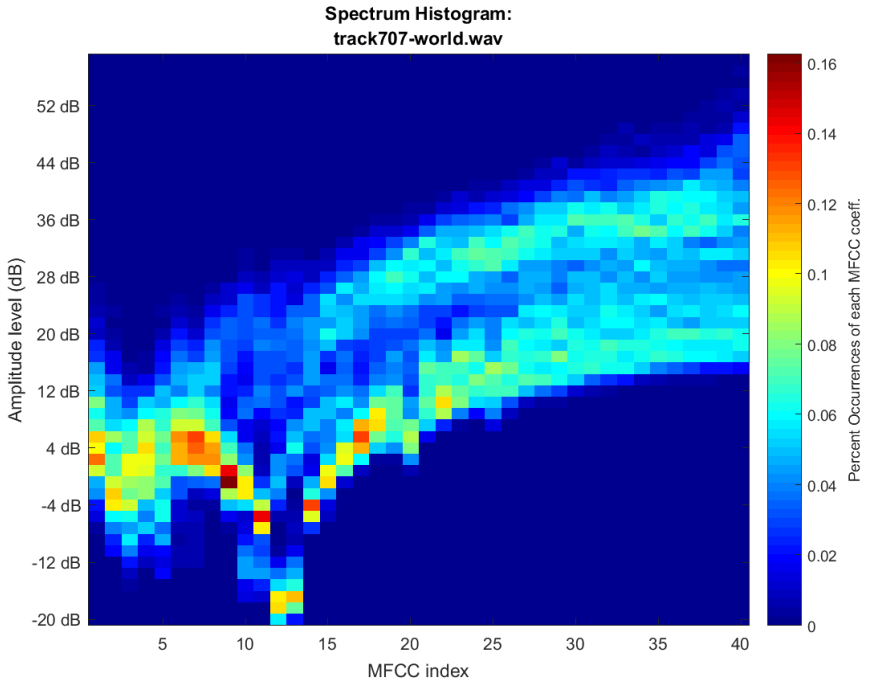

Figure 17: Spectrum Histogram 707

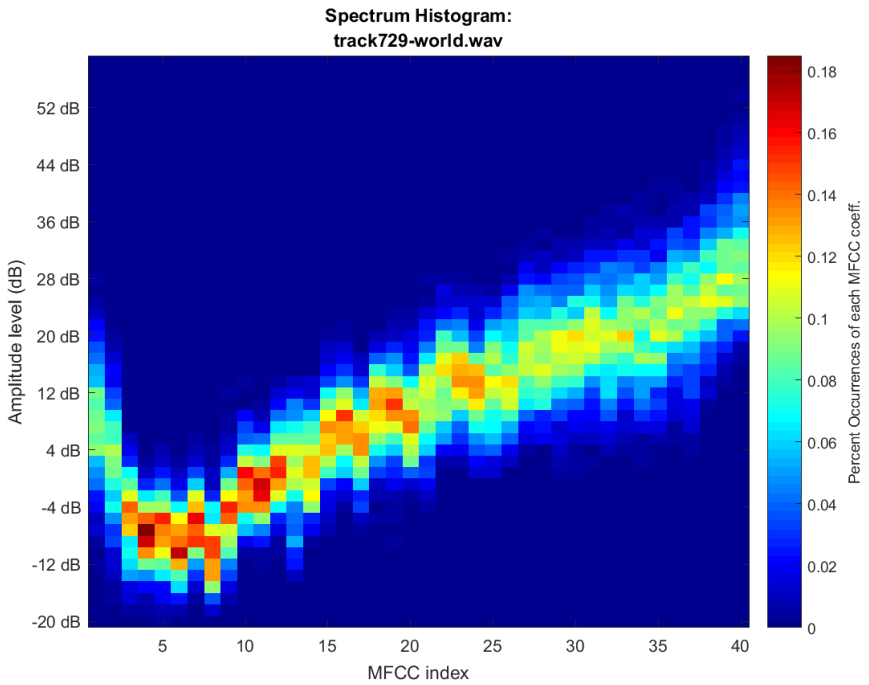

Figure 18: Spectrum Histogram 729

A.2 Similarity Matrix

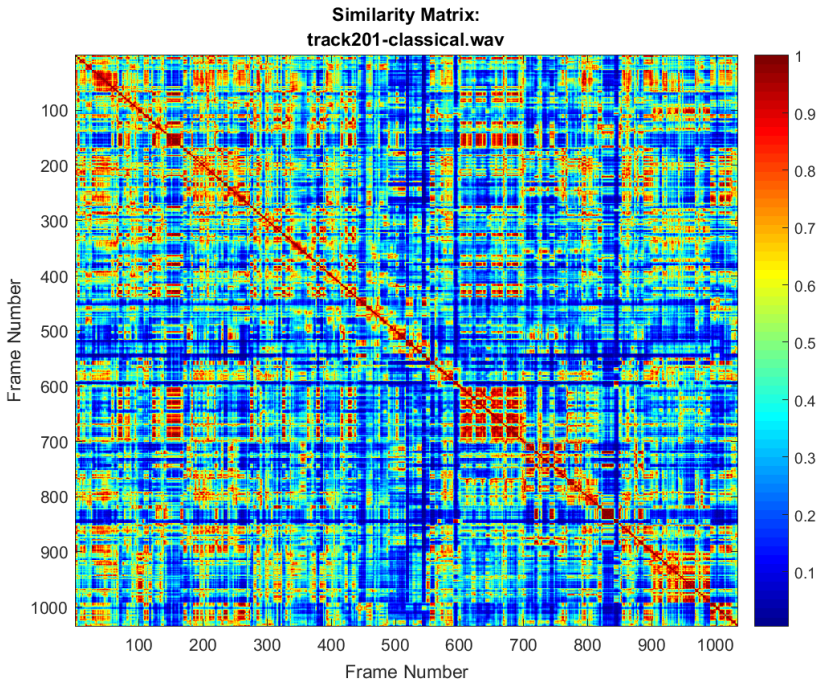

Figure 19: Similarity Matrix 201

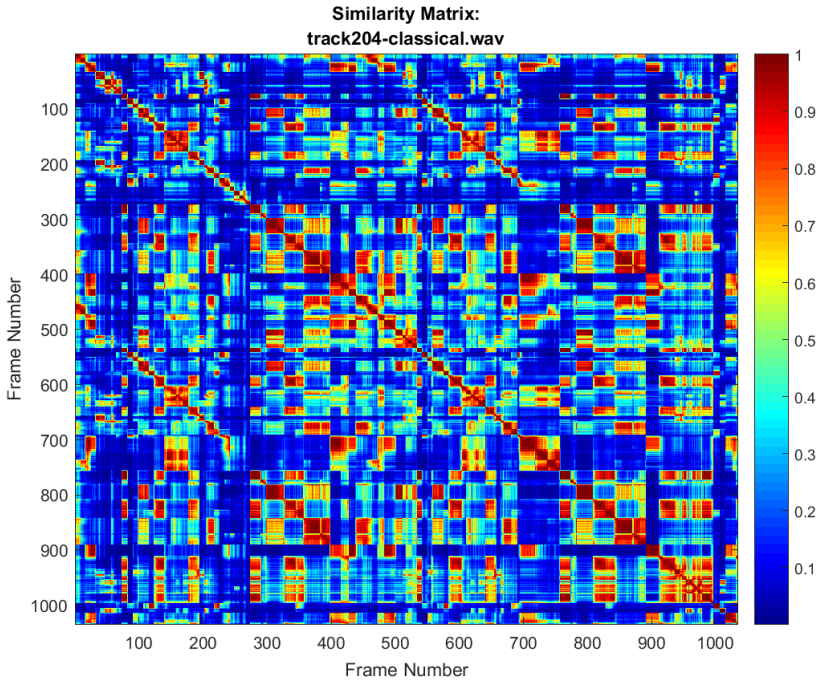

Figure 20: Similarity Matrix 204

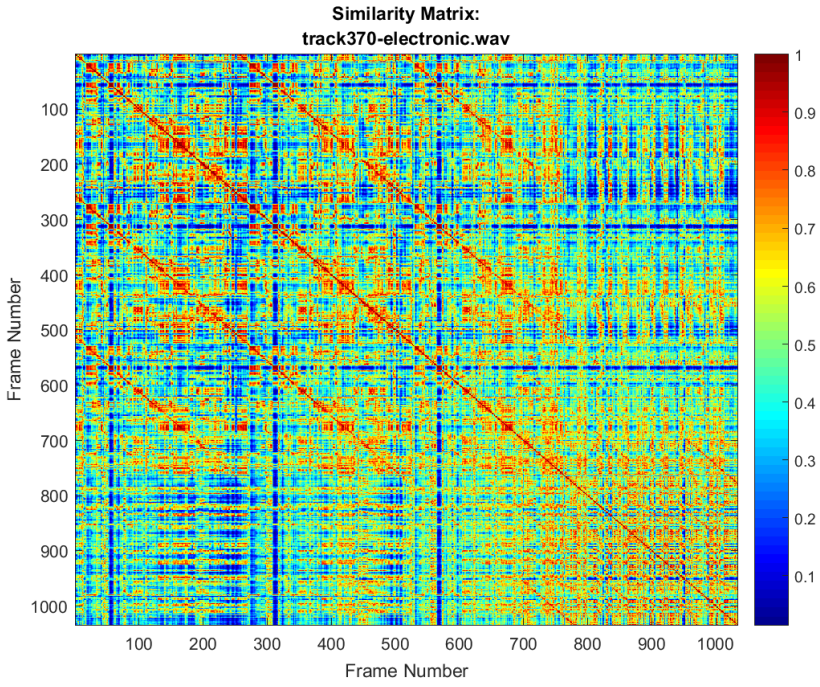

Figure 21: Similarity Matrix 370

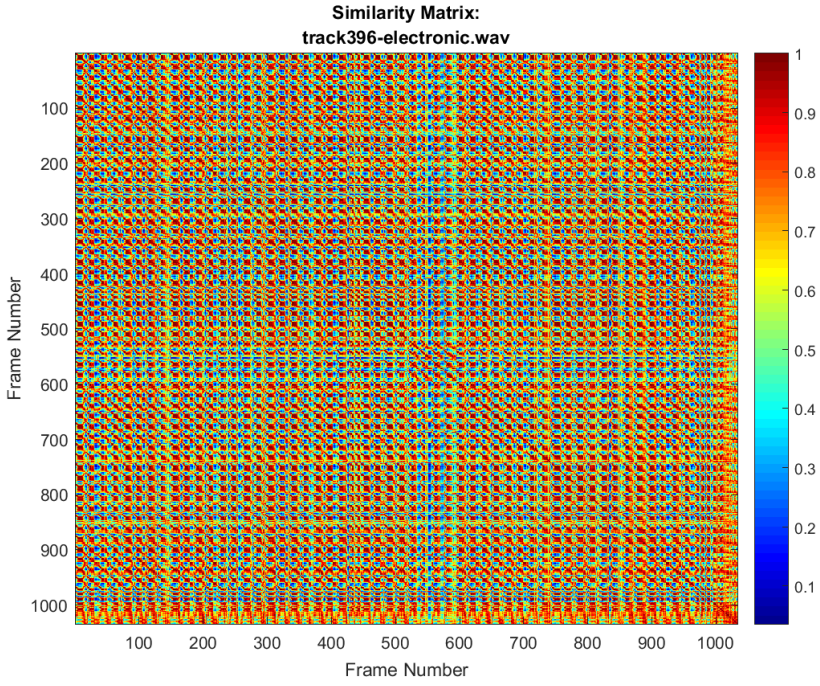

Figure 22: Similarity Matrix 396

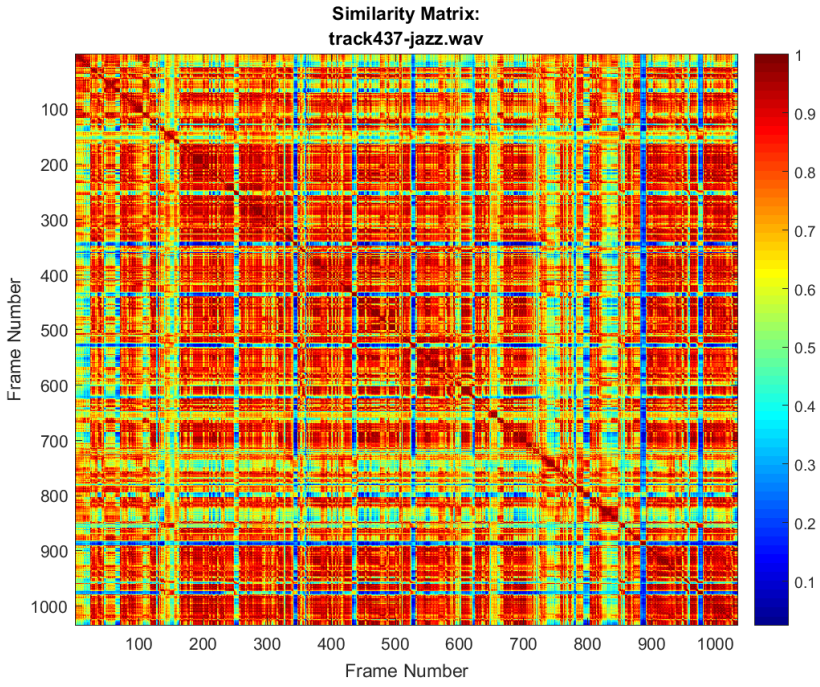

Figure 23: Similarity Matrix 437

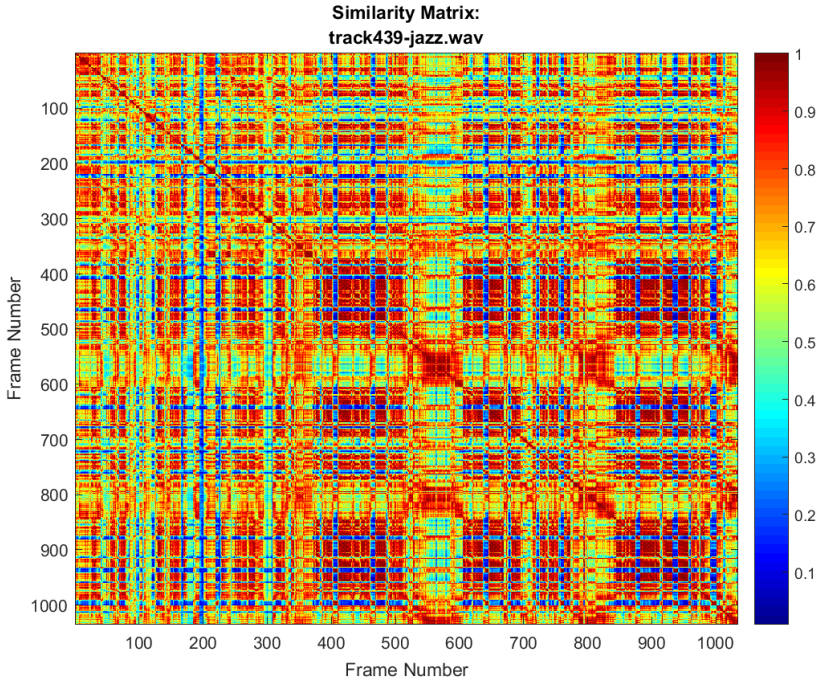

Figure 24: Similarity Matrix 439

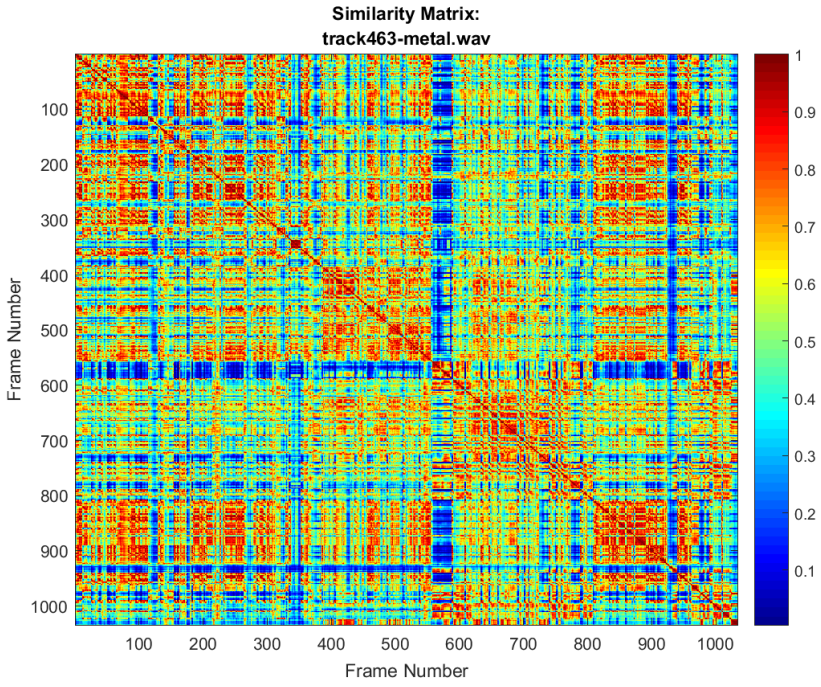

Figure 25: Similarity Matrix 463

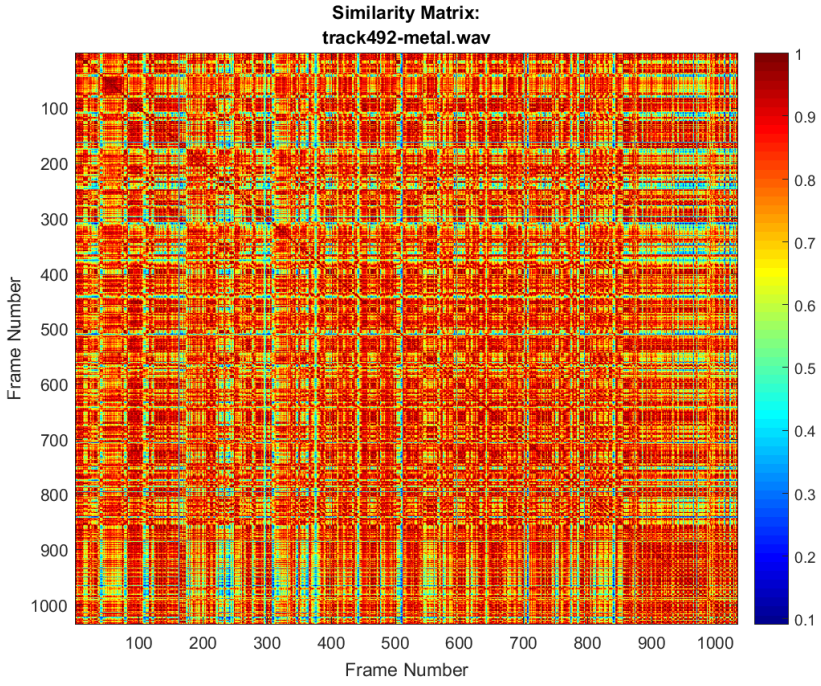

Figure 26: Similarity Matrix 492

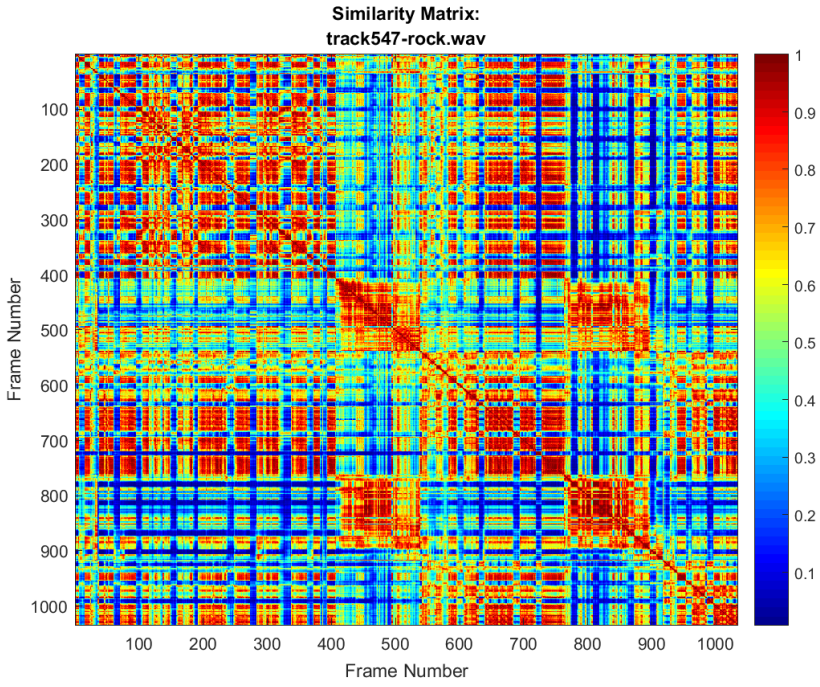

Figure 27: Similarity Matrix 547

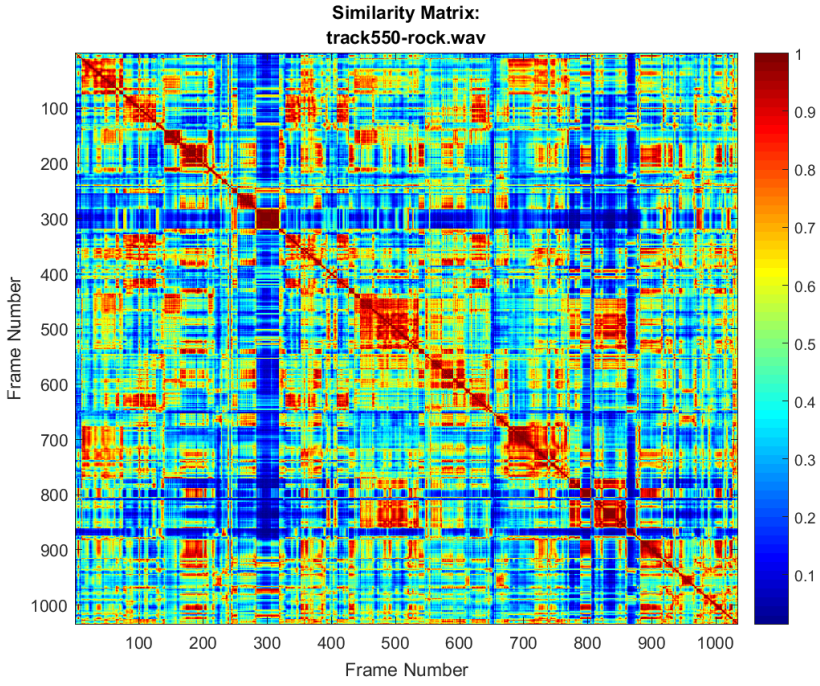

Figure 28: Similarity Matrix 550

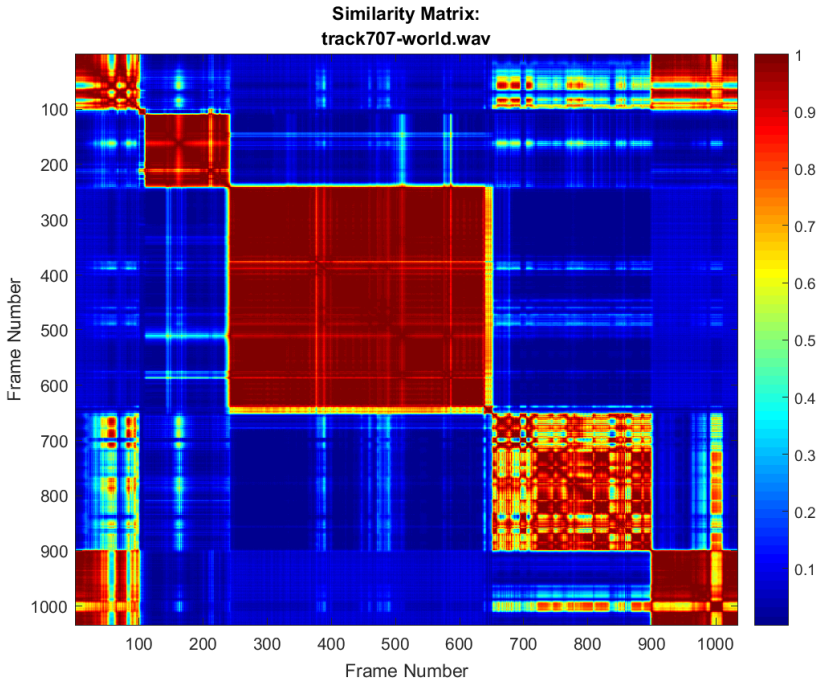

Figure 29: Similarity Matrix 707

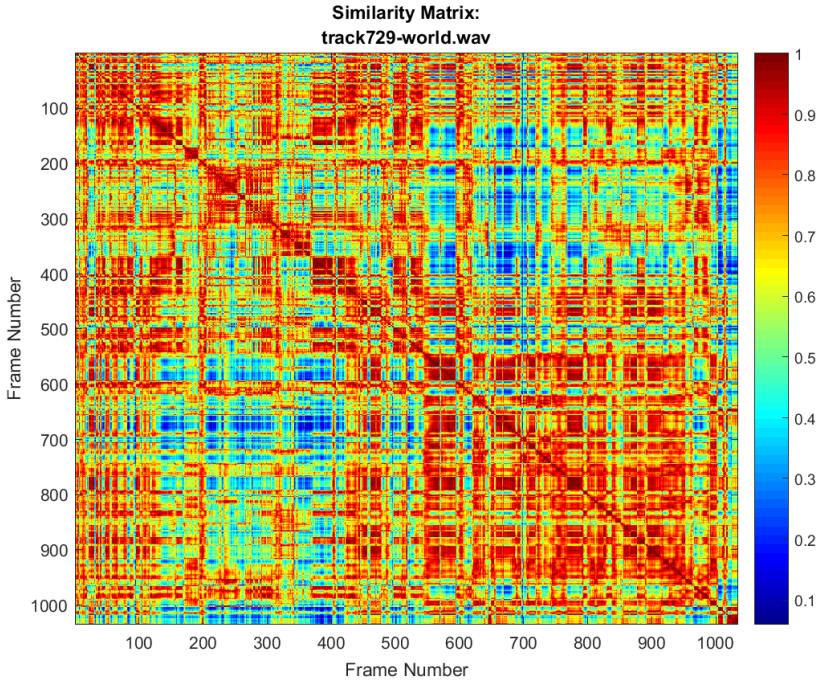

Figure 30: Similarity Matrix 729

A.3 Rhythm Index

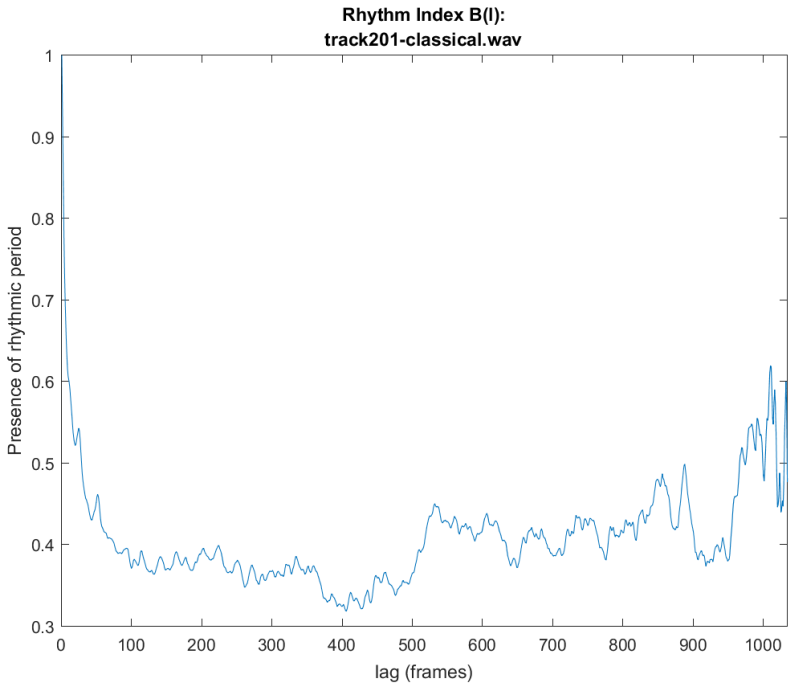

Figure 31: Rhythm Index 201

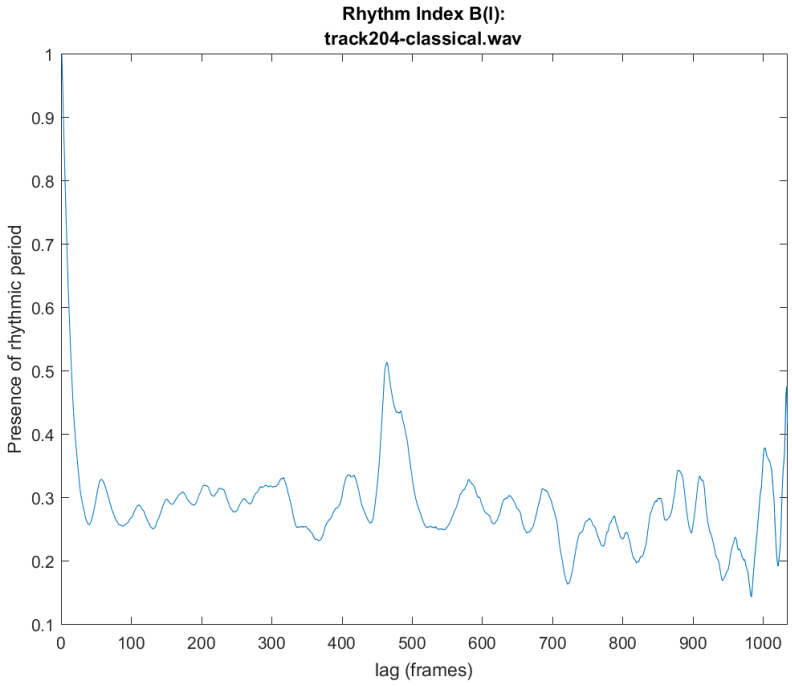

Figure 32: Rhythm Index 204

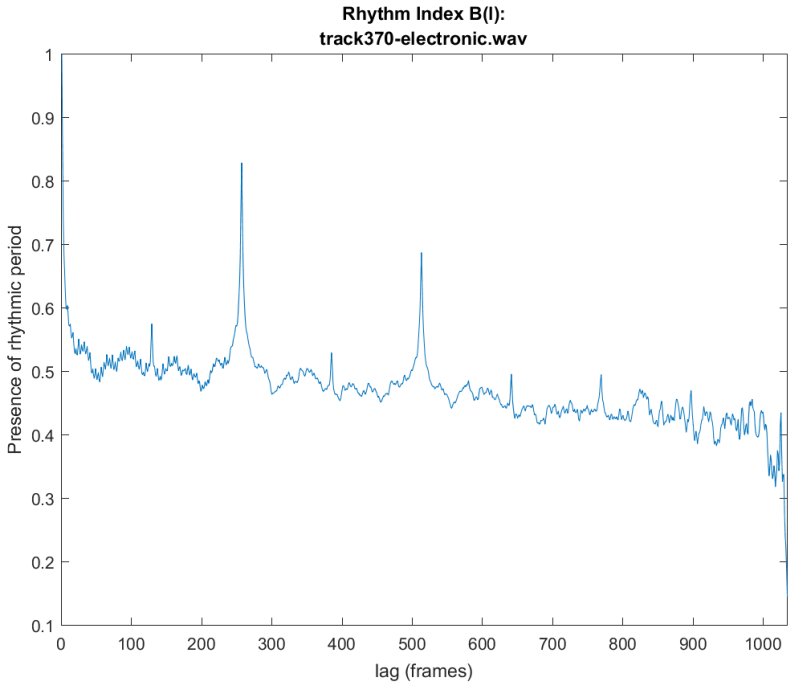

Figure 33: Rhythm Index 370

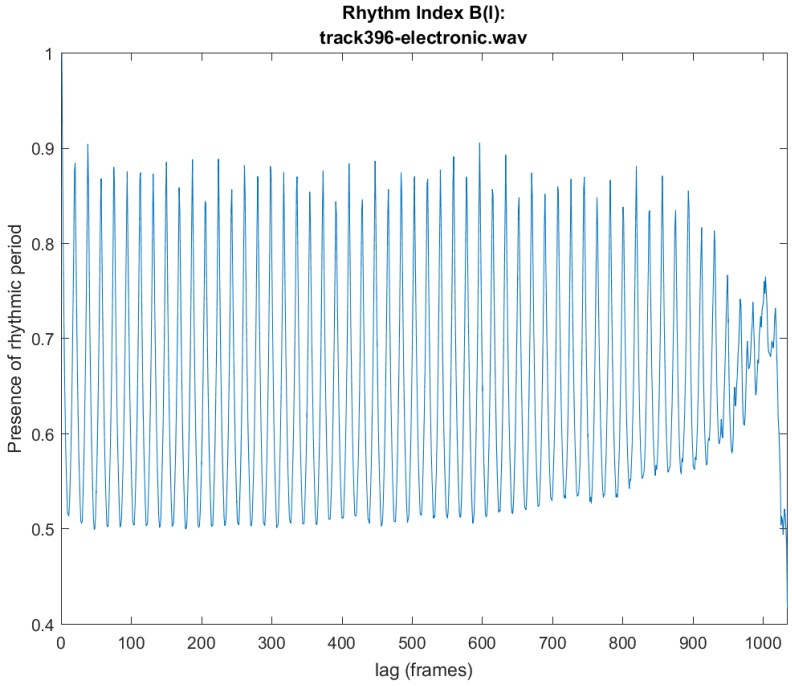

Figure 34: Rhythm Index 396

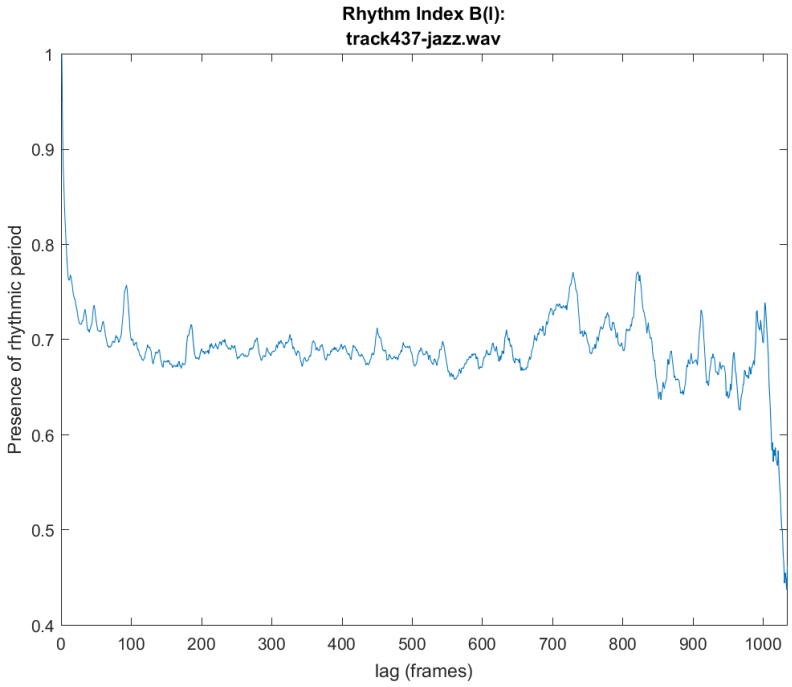

Figure 35: Rhythm Index 437

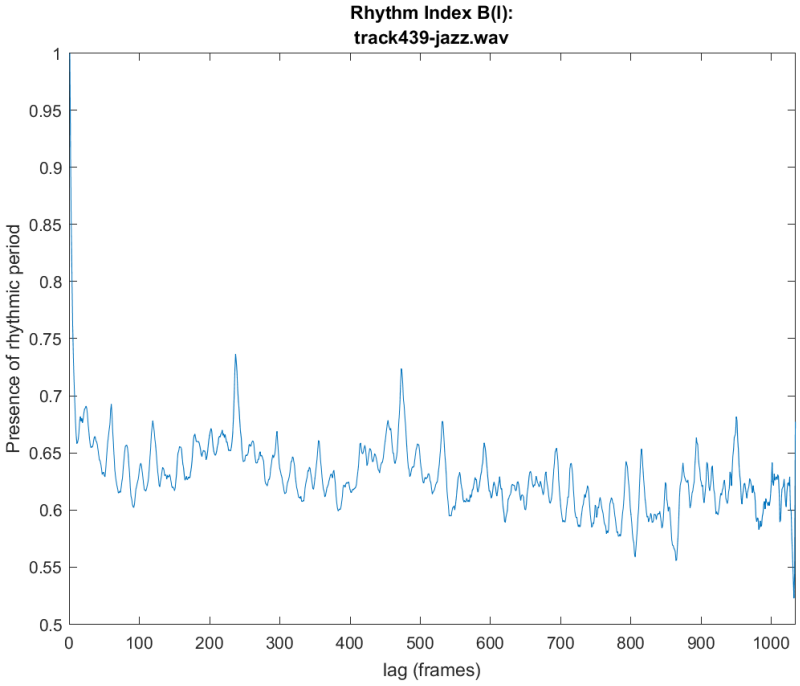

Figure 36: Rhythm Index 439

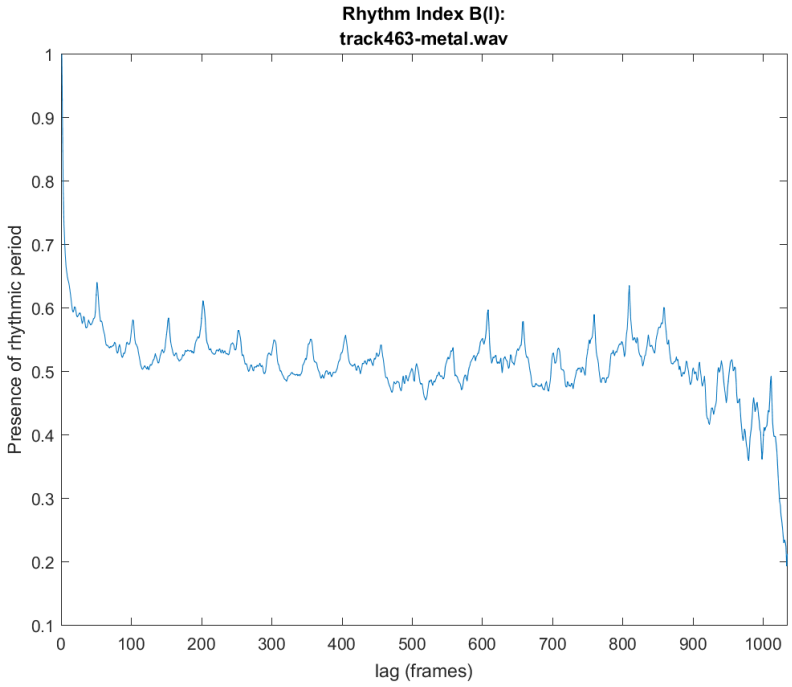

Figure 37: Rhythm Index 463

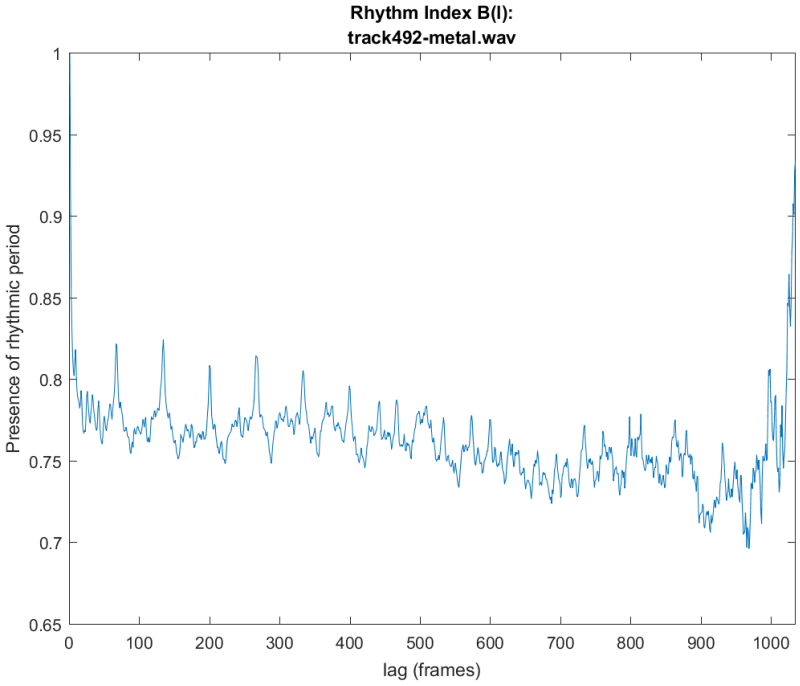

Figure 38: Rhythm Index 492

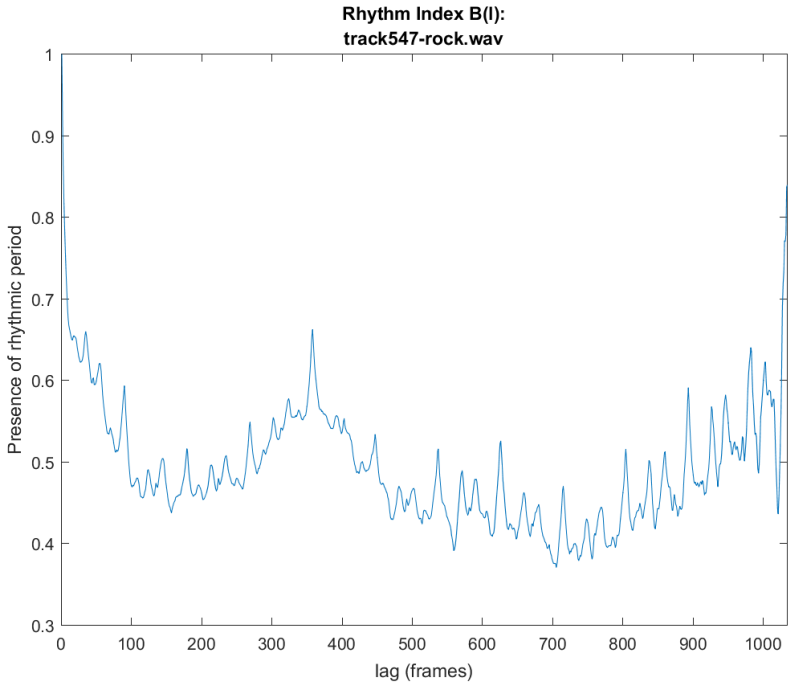

Figure 39: Rhythm Index 547

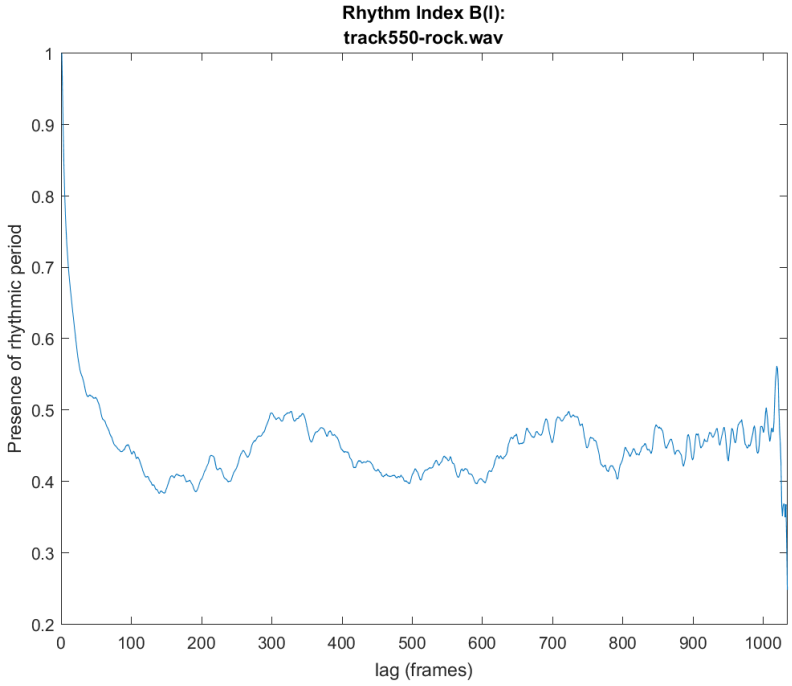

Figure 40: Rhythm Index 550

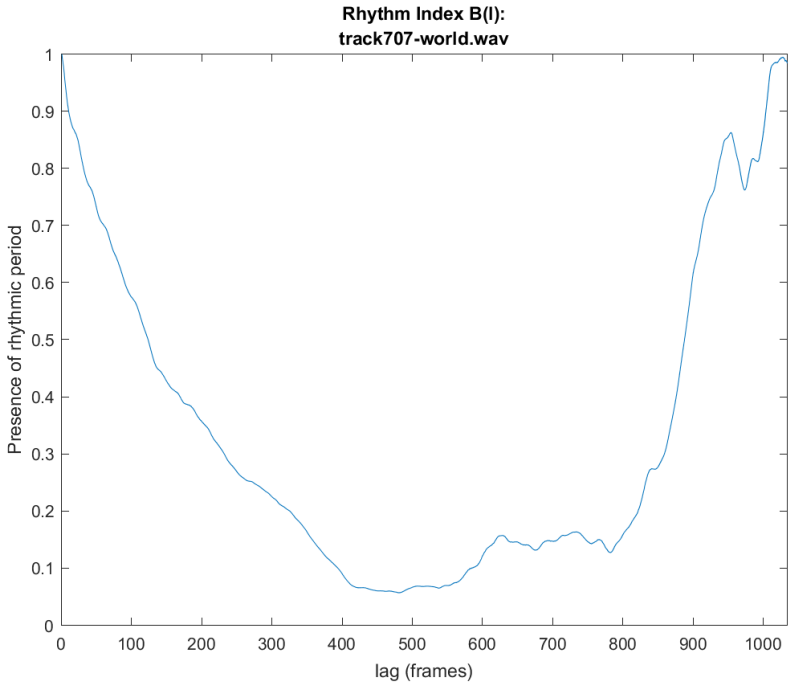

Figure 41: Rhythm Index 707

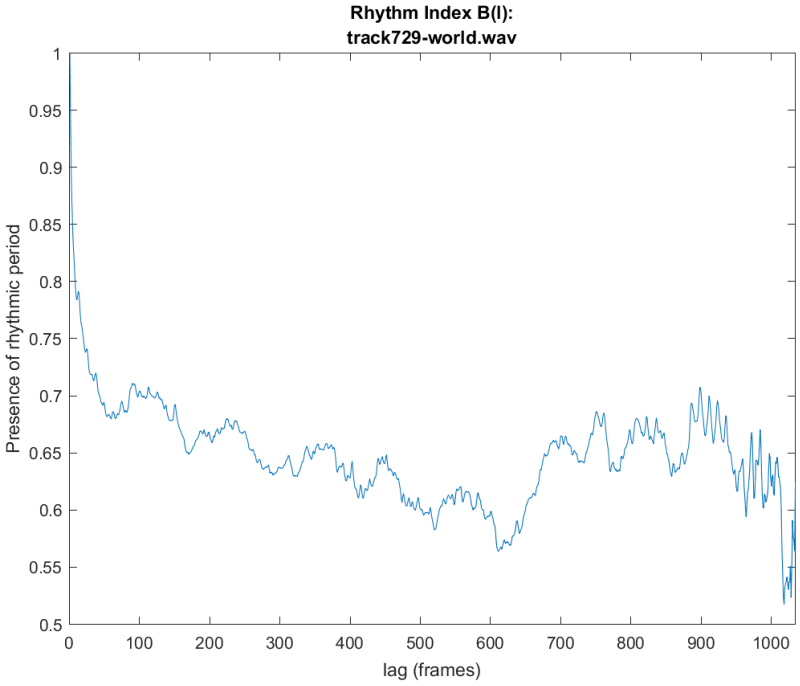

Figure 42: Rhythm Index 729

A.4 Autocorrelation

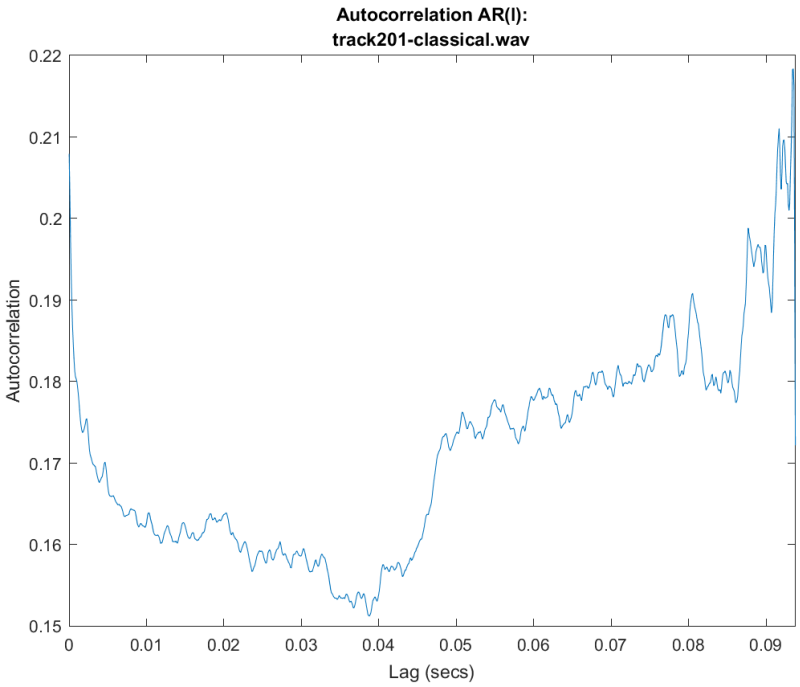

Figure 43: Autocorrelation 201

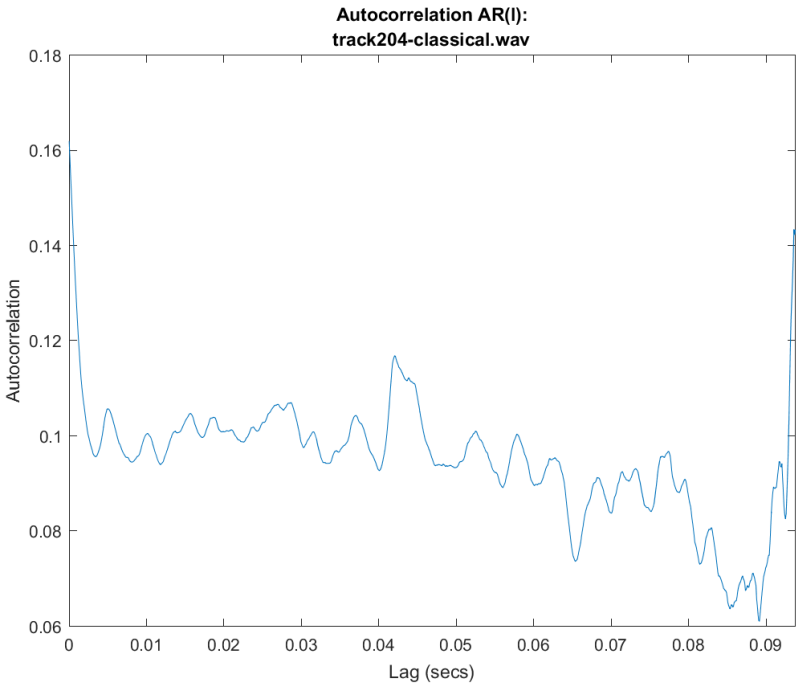

Figure 44: Autocorrelation 204

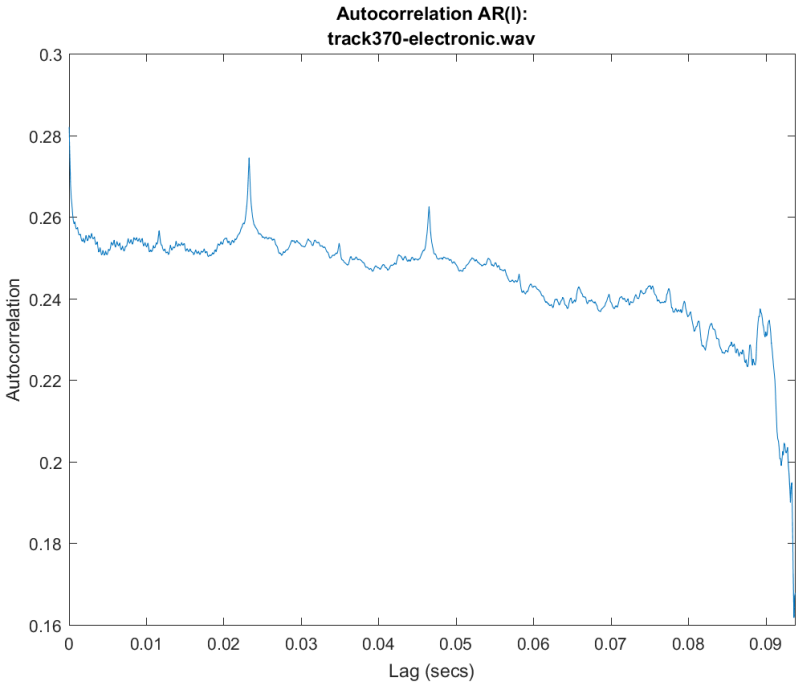

Figure 45: Autocorrelation 370

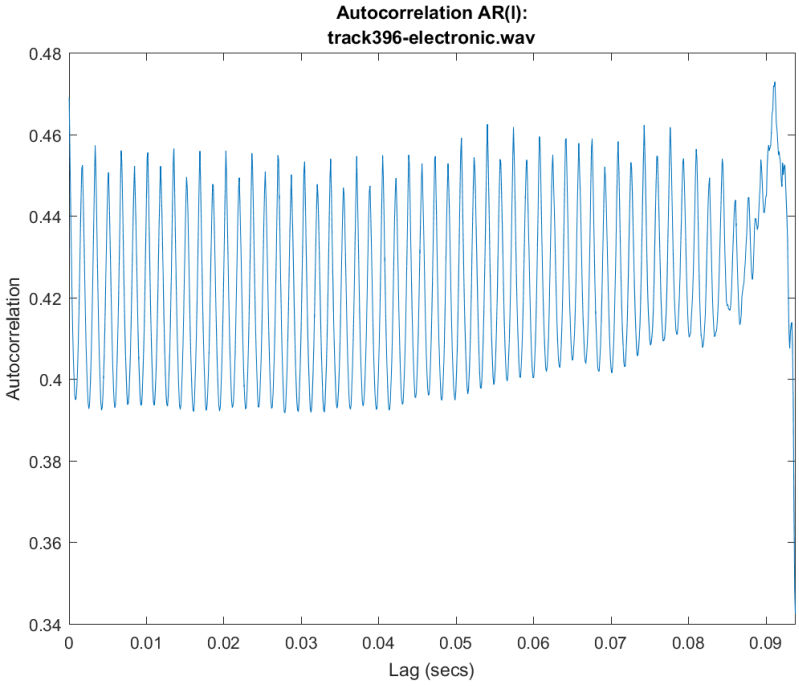

Figure 46: Autocorrelation 396

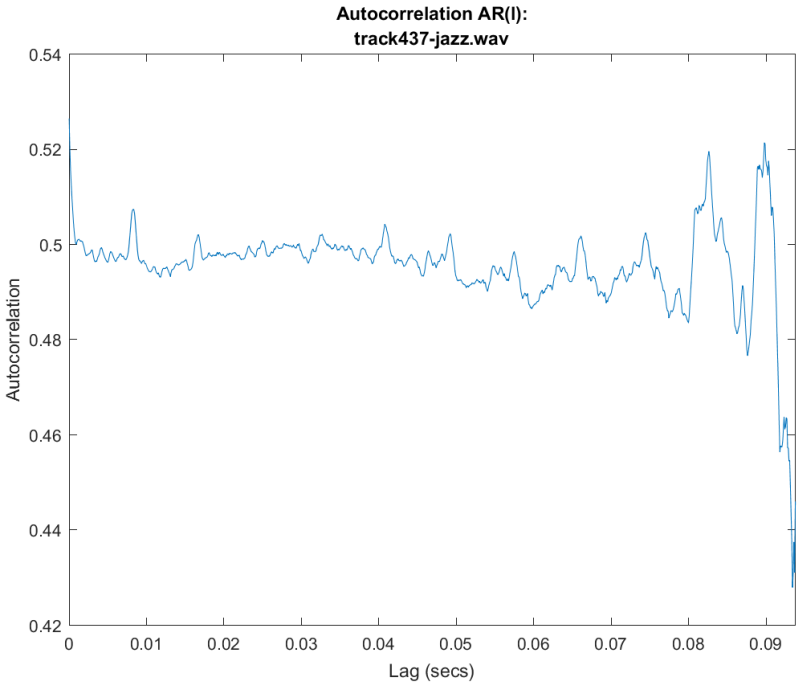

Figure 47: Autocorrelation 437

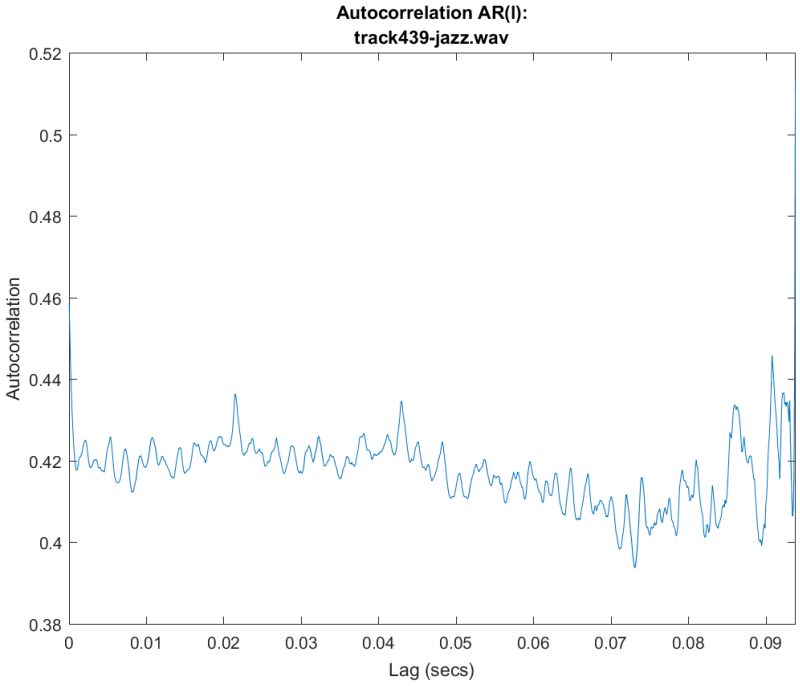

Figure 48: Autocorrelation 439

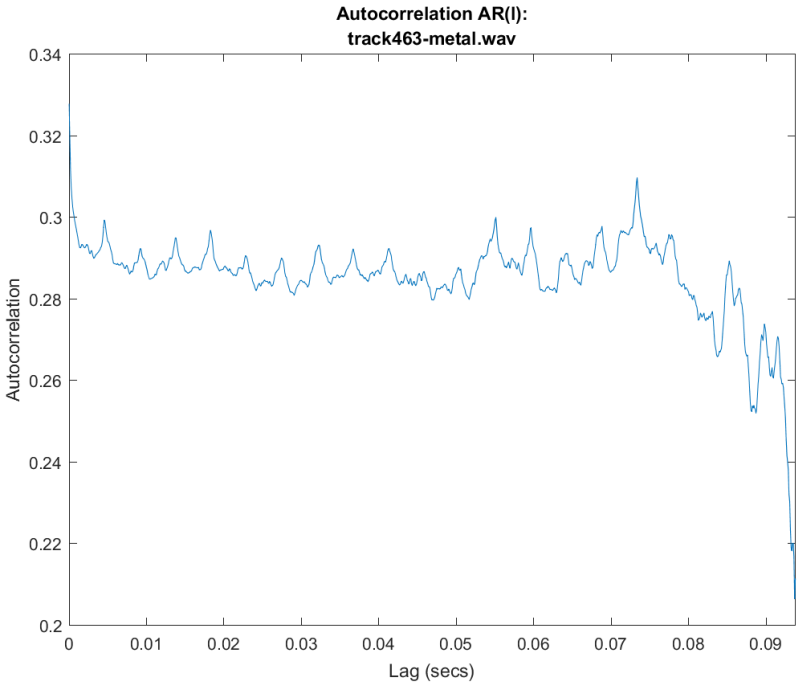

Figure 49: Autocorrelation 463

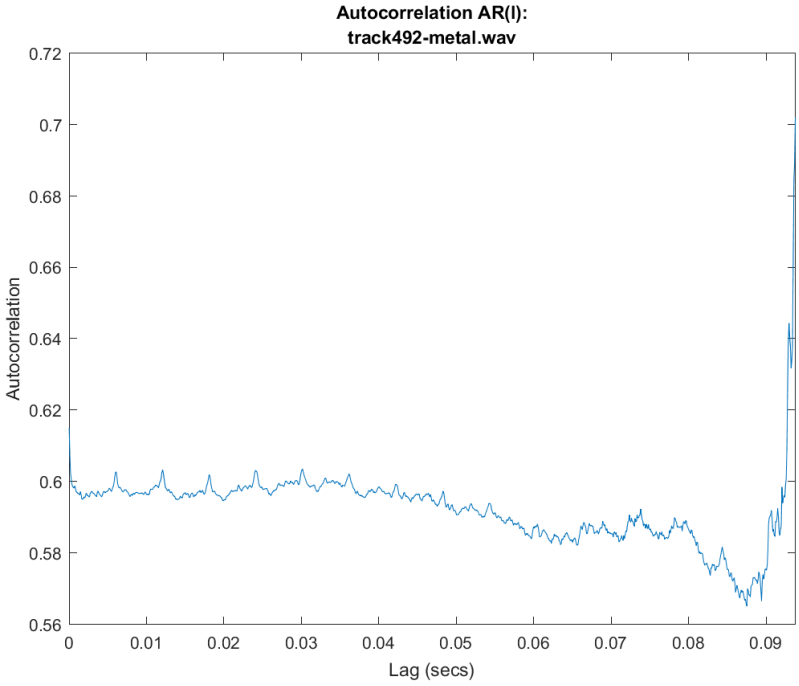

Figure 50: Autocorrelation 492

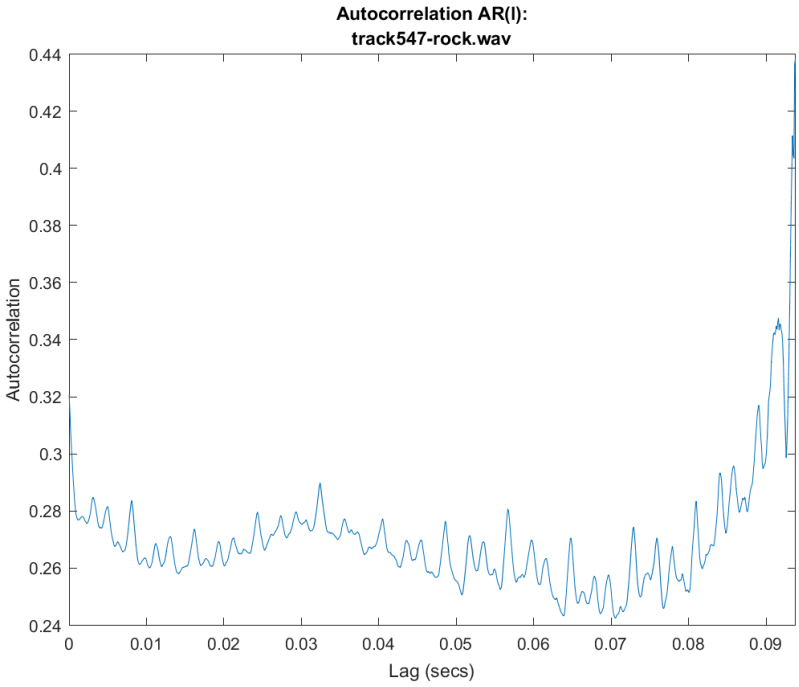

Figure 51: Autocorrelation 547

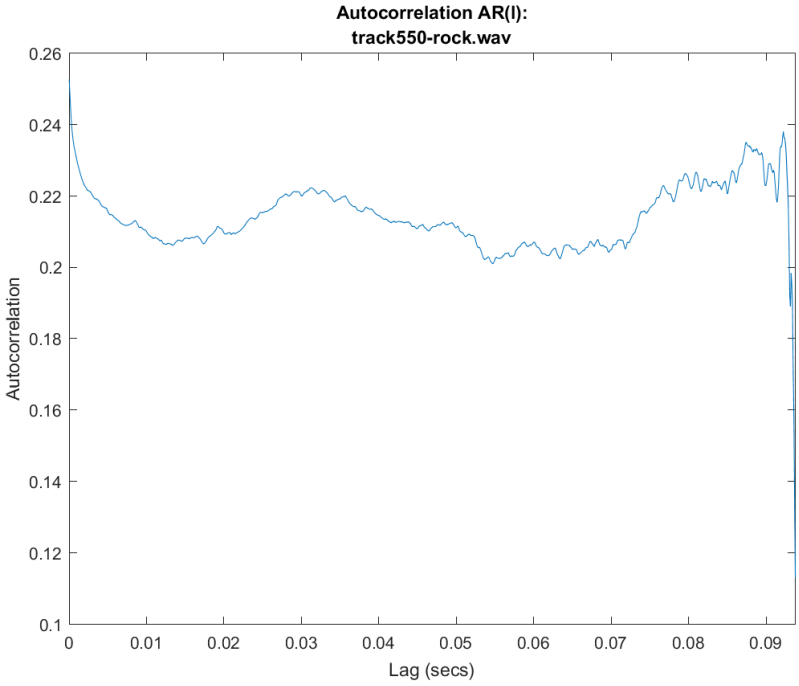

Figure 52: Autocorrelation 550

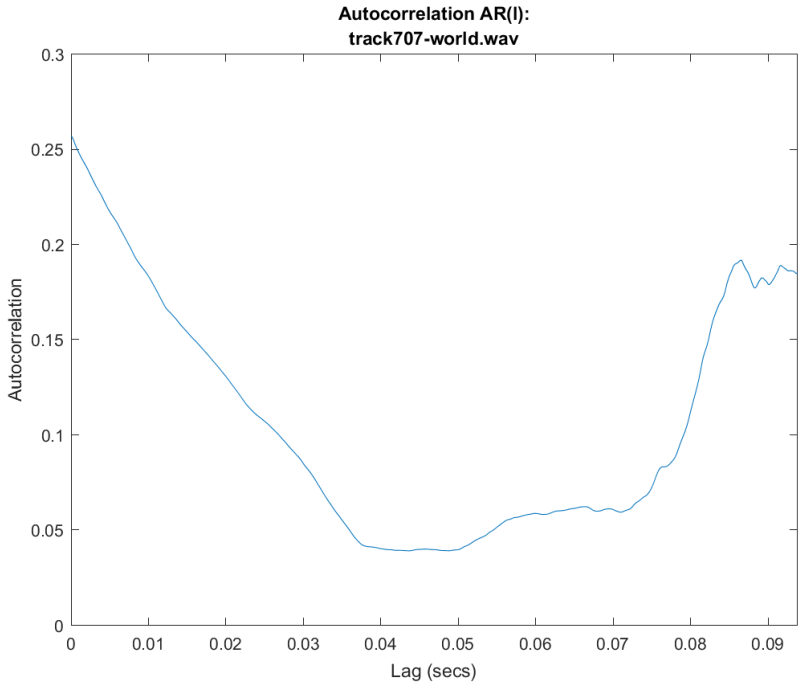

Figure 53: Autocorrelation 707

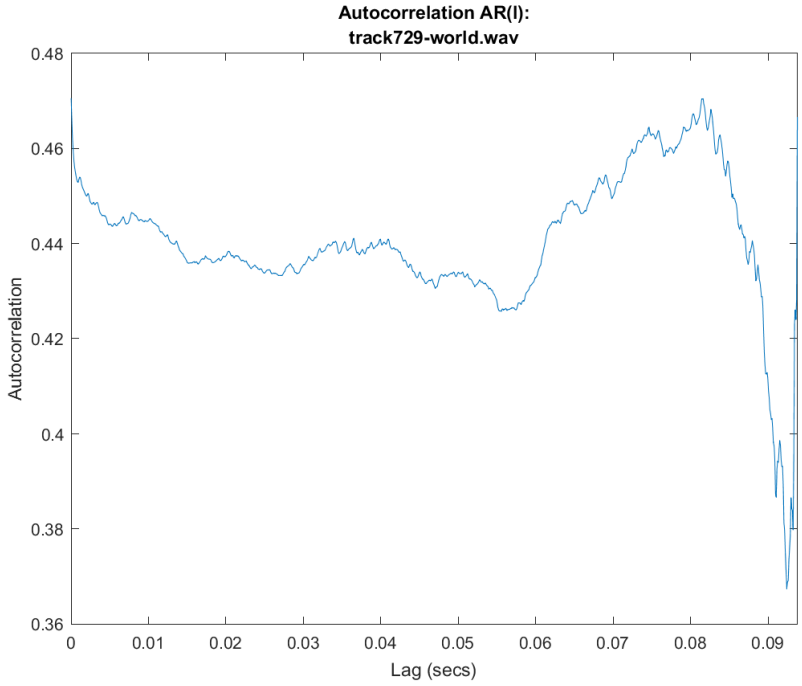

Figure 54: Autocorrelation 729

A.5 Dynamic Rhythmic Variation

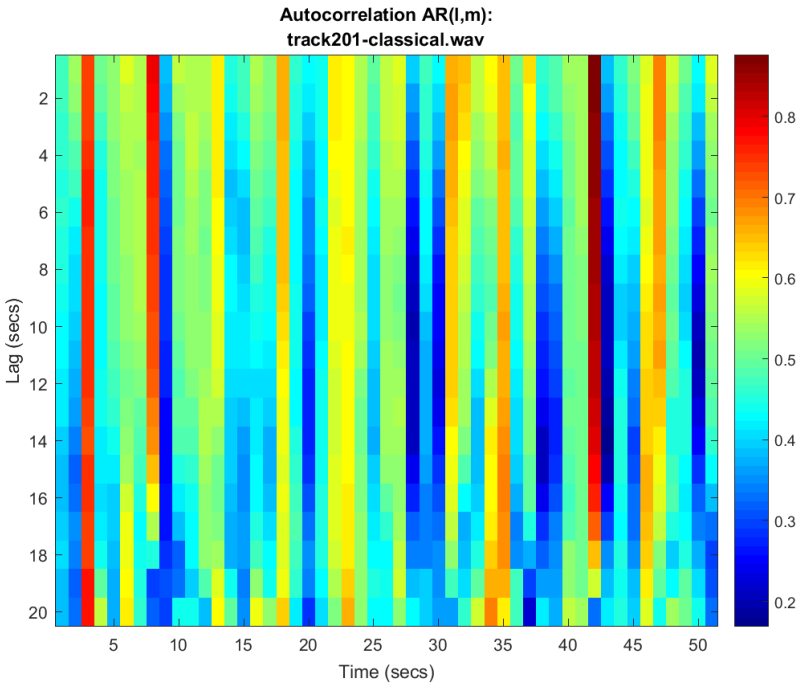

Figure 55: Rhythmic Variation 201

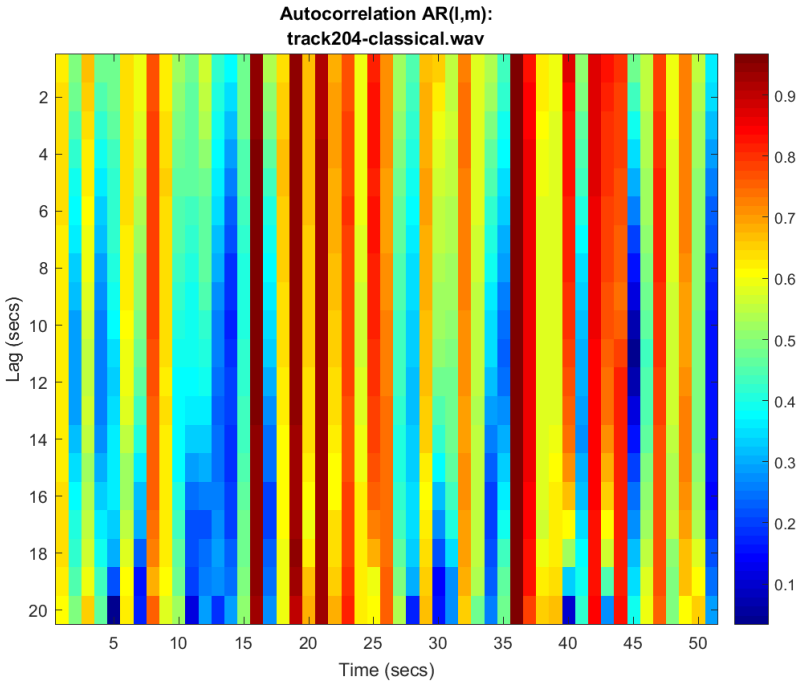

Figure 56: Rhythmic Variation 204

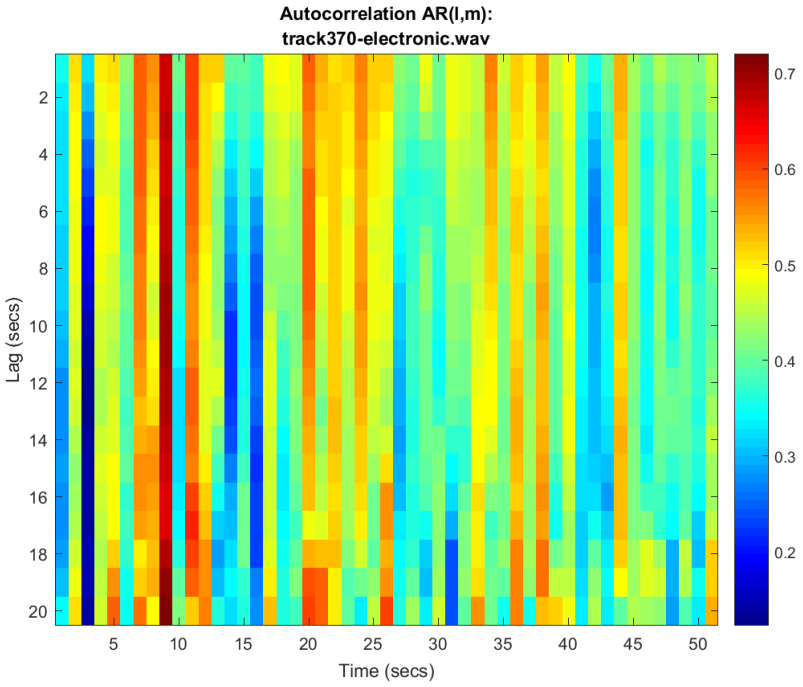

Figure 57: Rhythmic Variation 370

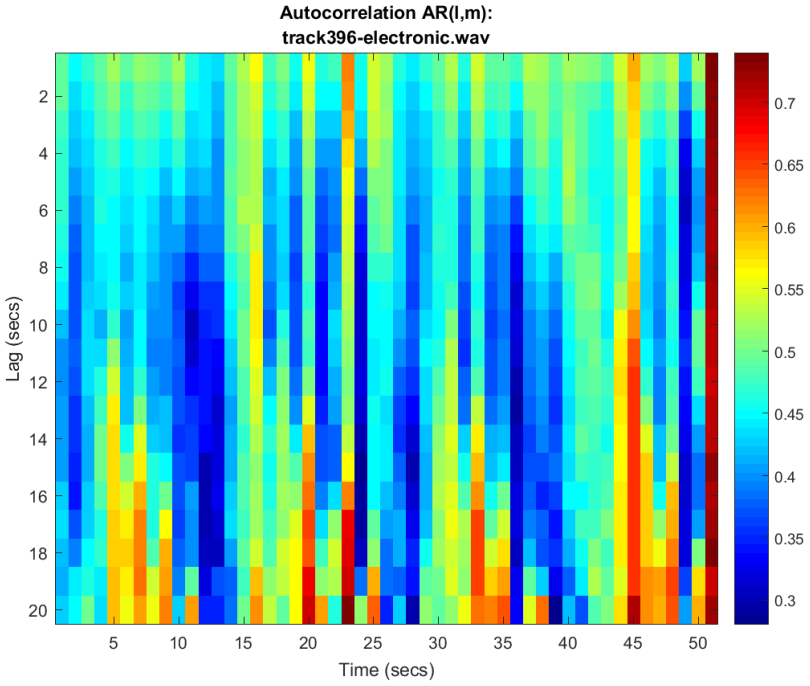

Figure 58: Rhythmic Variation 396

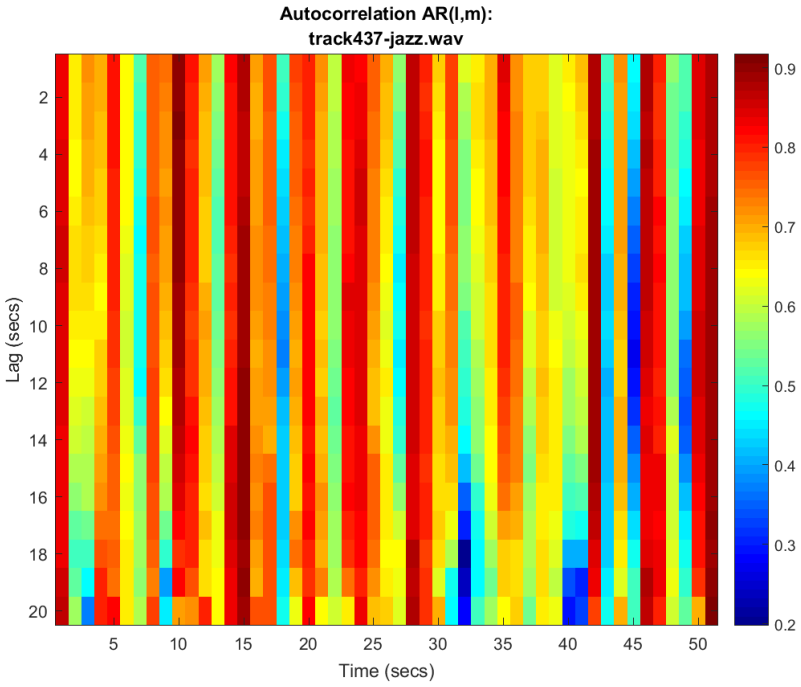

Figure 59: Rhythmic Variation 437

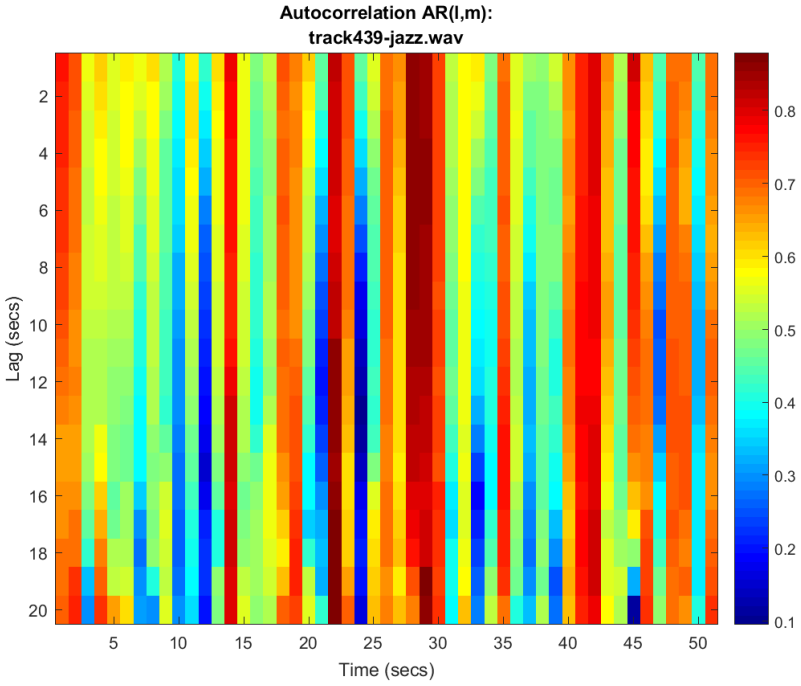

Figure 60: Rhythmic Variation 439

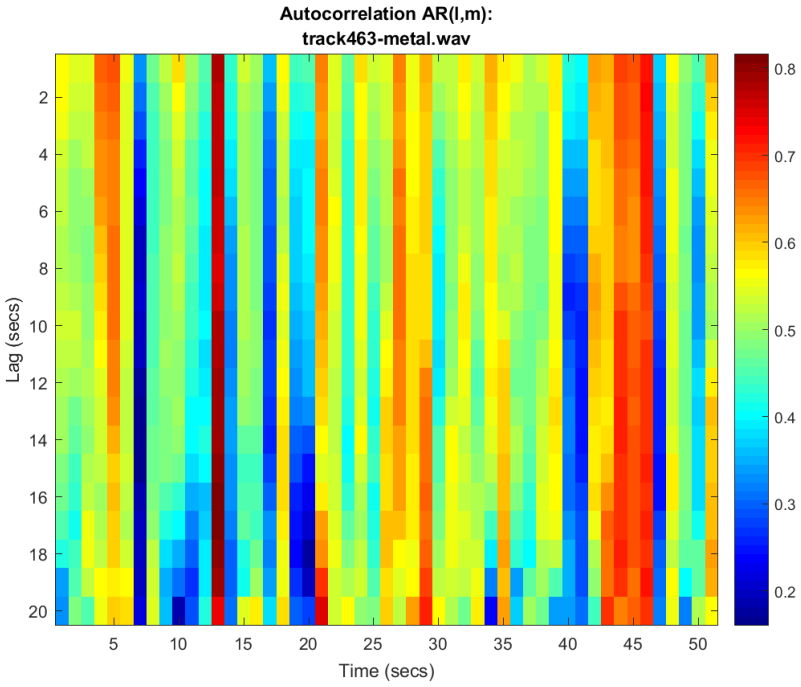

Figure 61: Rhythmic Variation 463

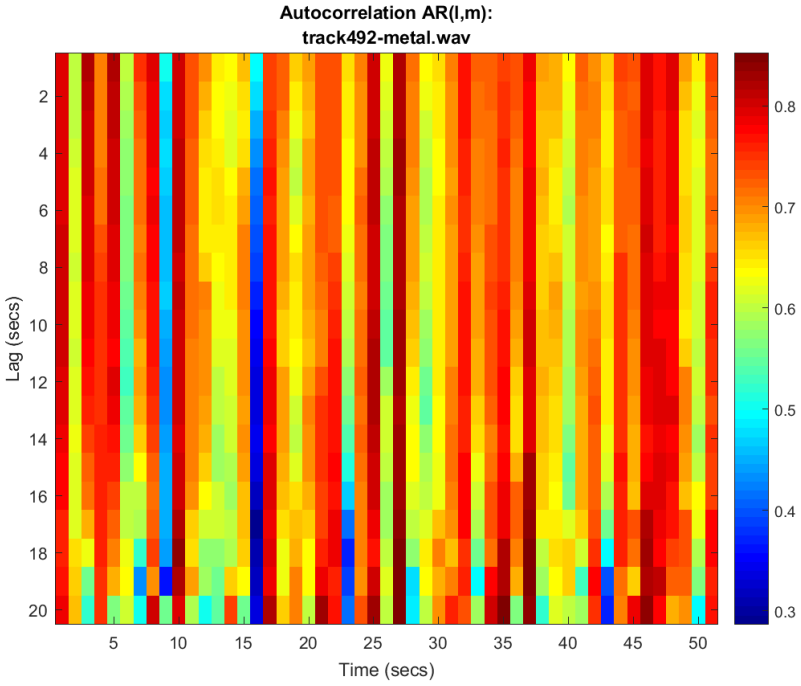

Figure 62: Rhythmic Variation 492

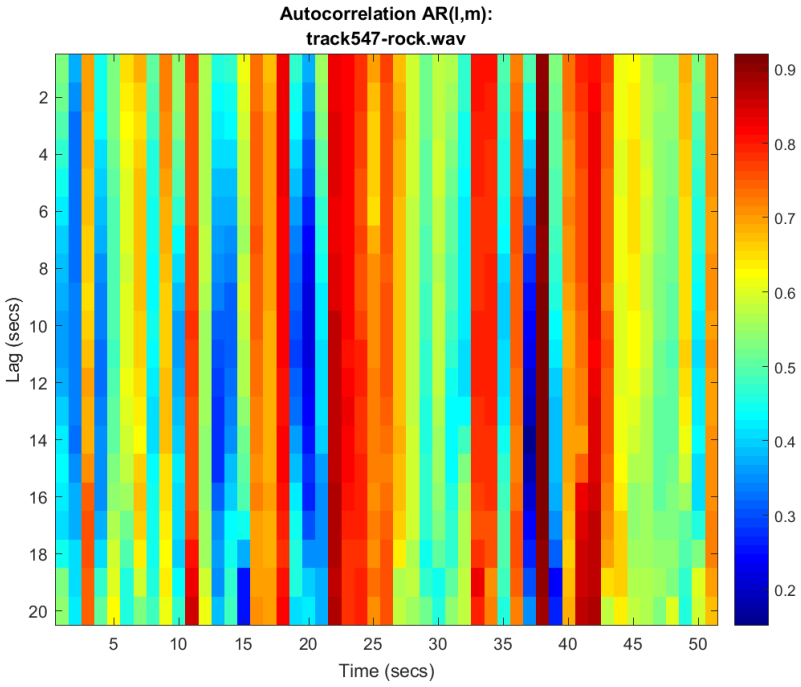

Figure 63: Rhythmic Variation 547

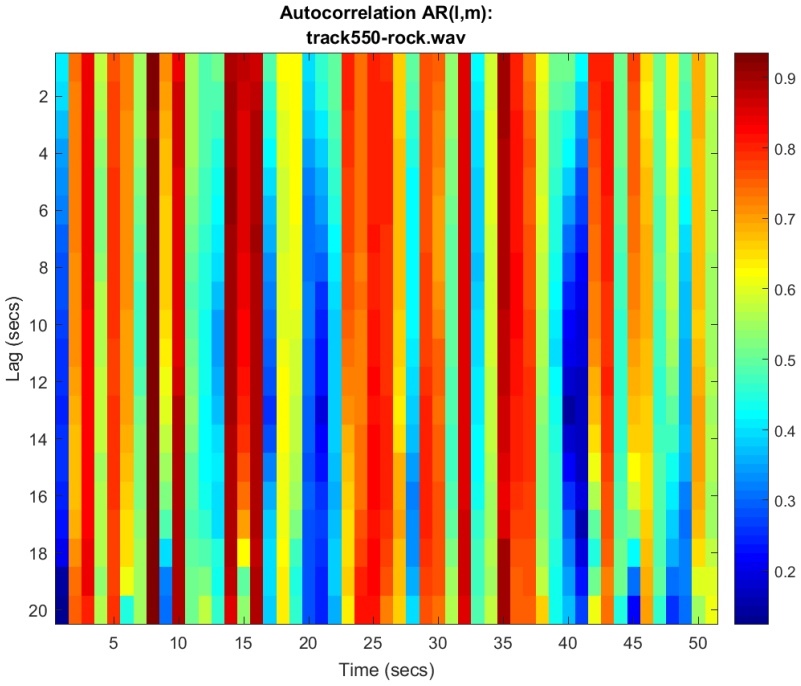

Figure 64: Rhythmic Variation 550

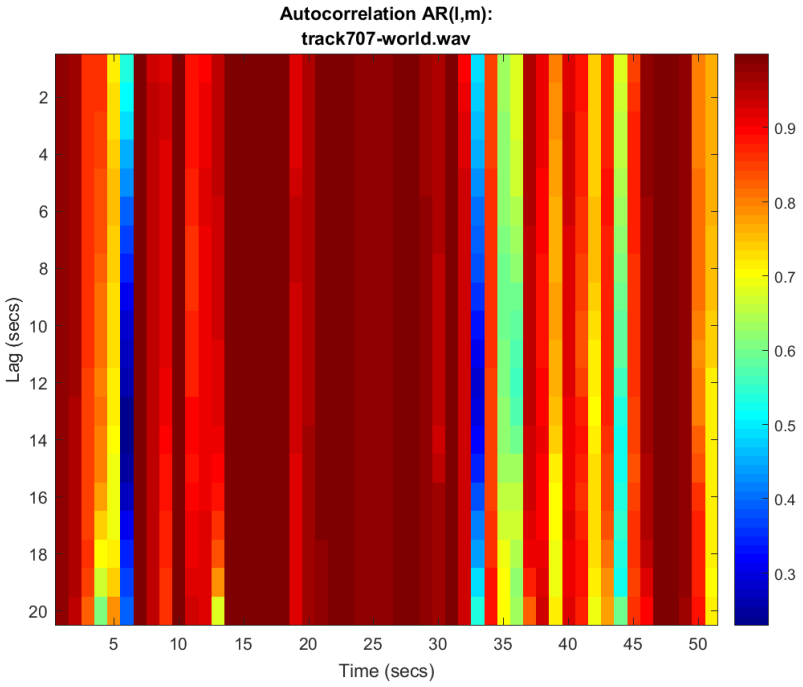

Figure 65: Rhythmic Variation 707

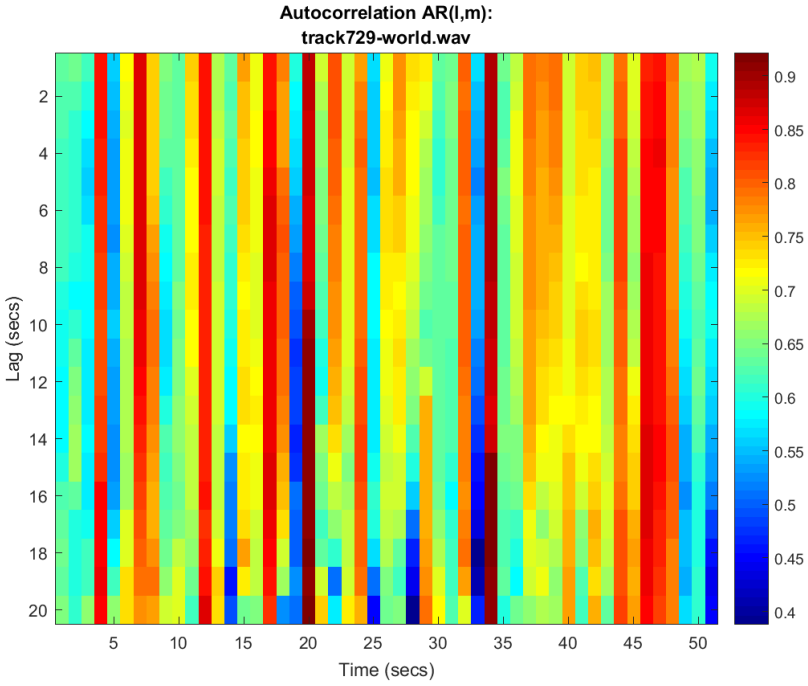

Figure 66: Rhythmic Variation 729

A.6 Pitch Class Profile

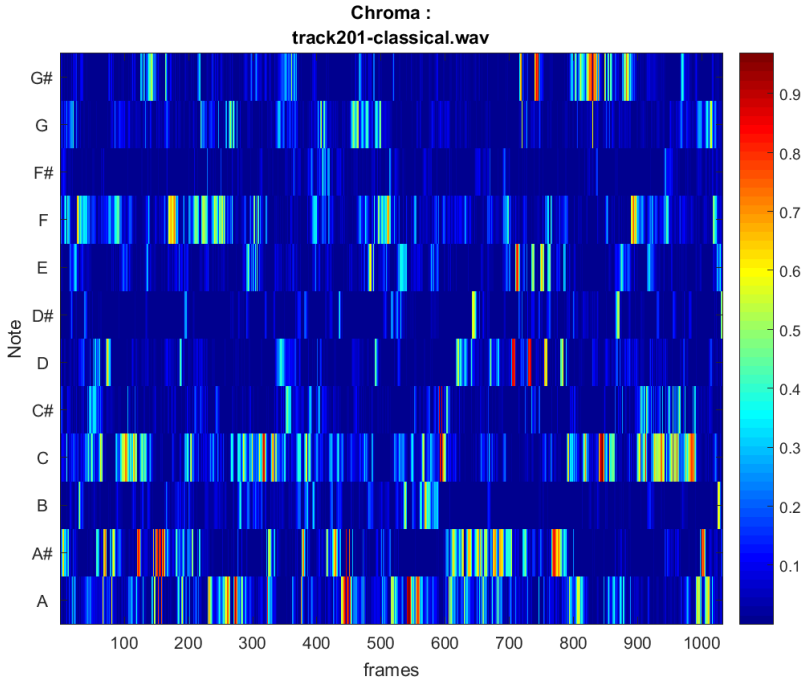

Figure 67: Pitch Class Profile 201

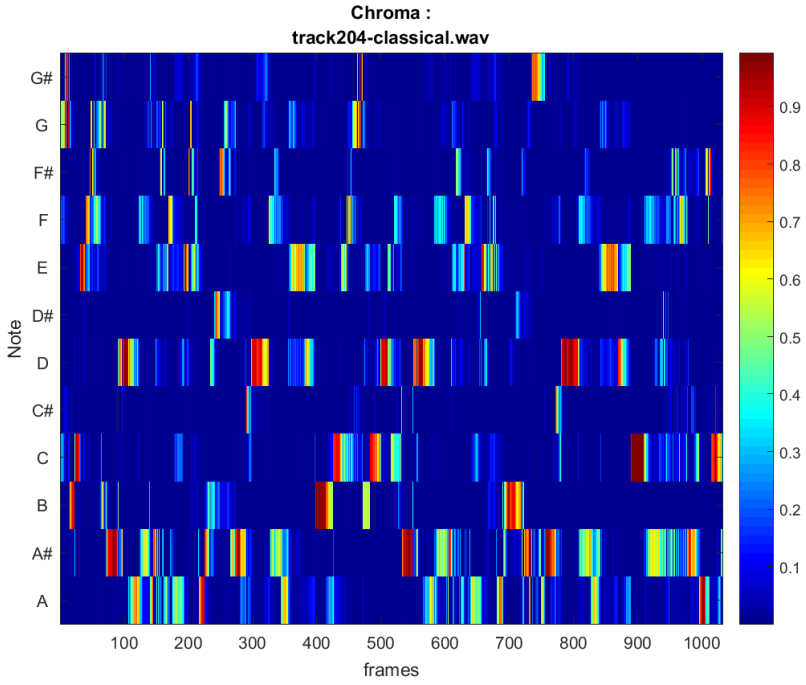

Figure 68: Pitch Class Profile 204

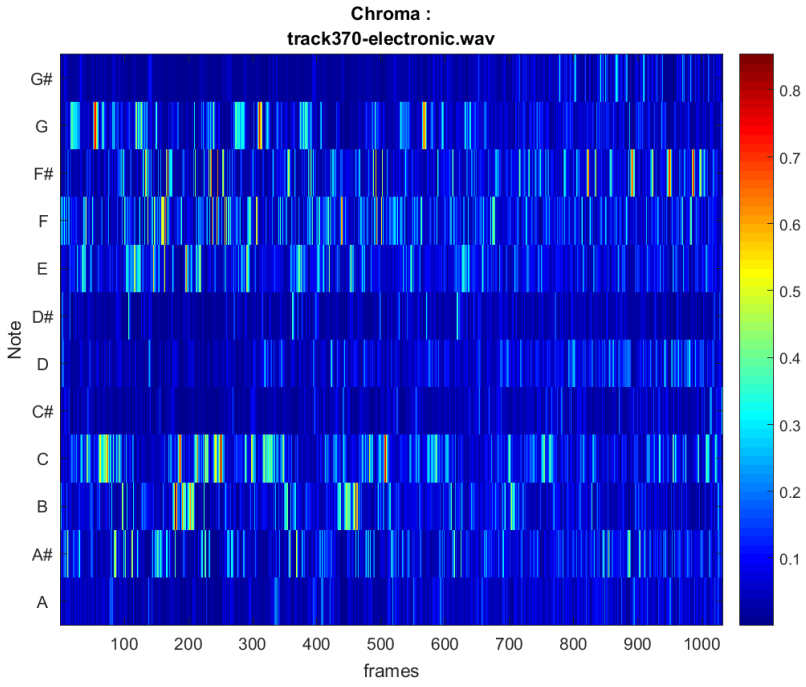

Figure 69: Pitch Class Profile 370

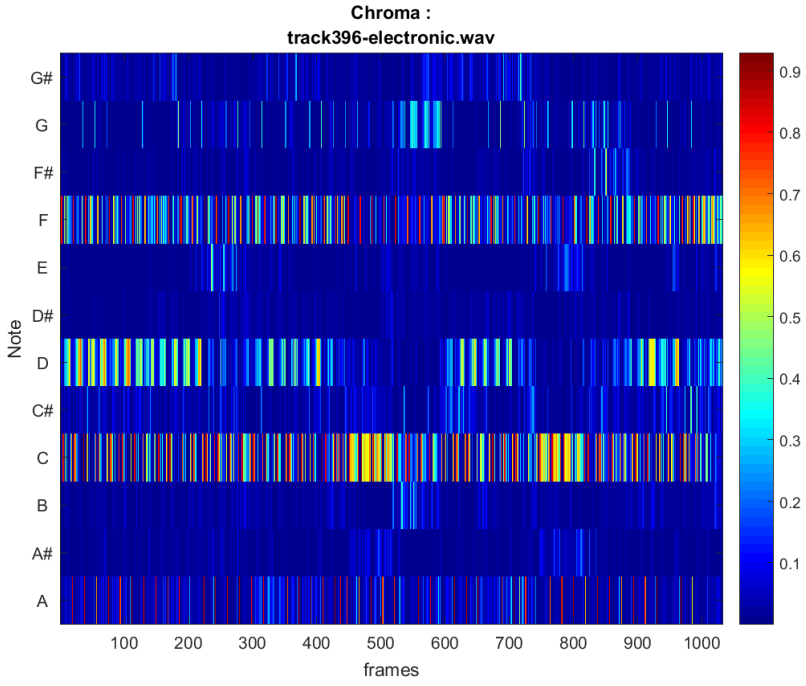

Figure 70: Pitch Class Profile 396

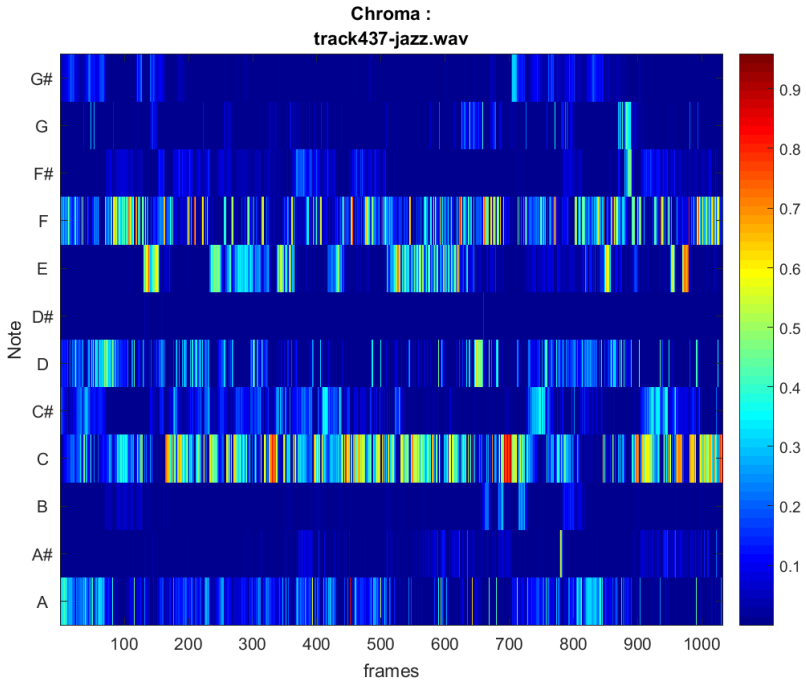

Figure 71: Pitch Class Profile 437

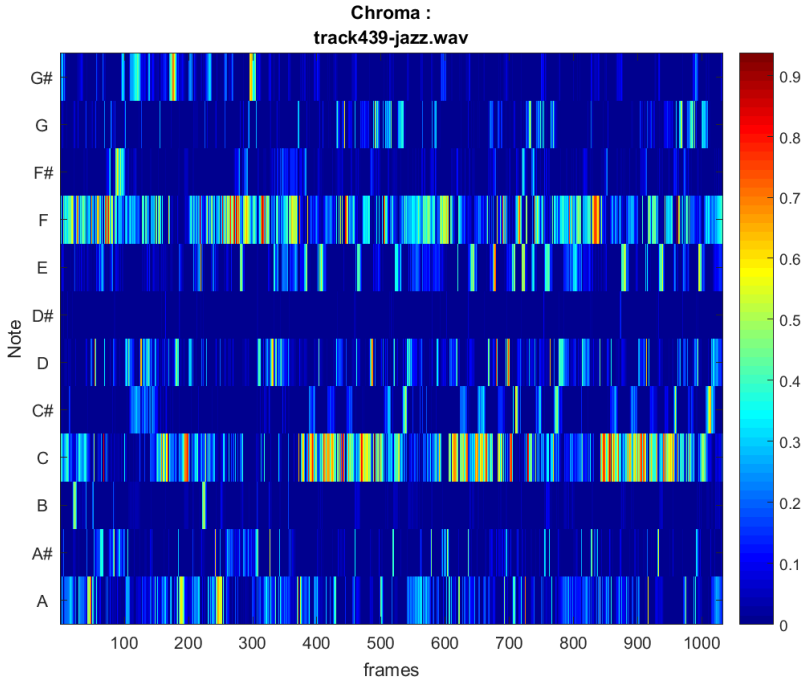

Figure 72: Pitch Class Profile 439

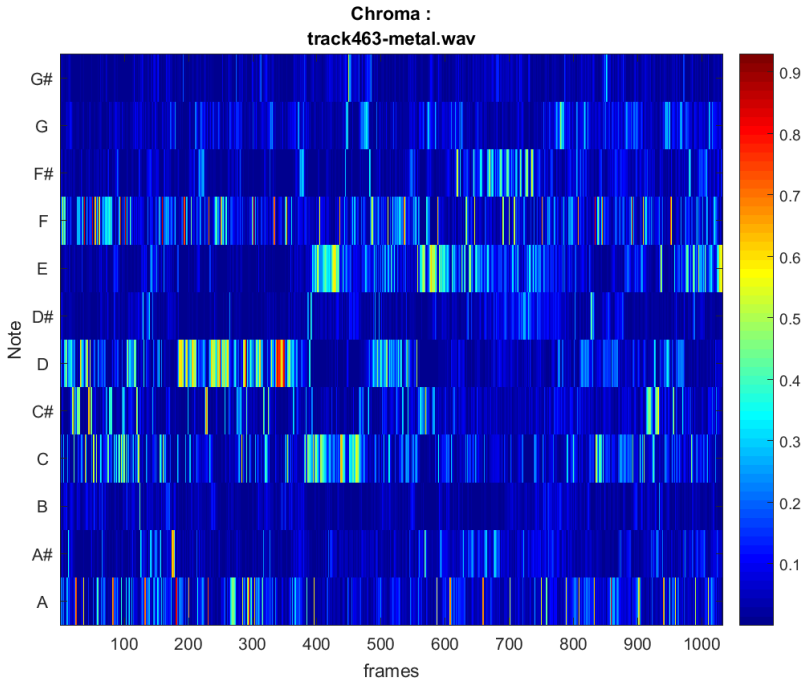

Figure 73: Pitch Class Profile 463

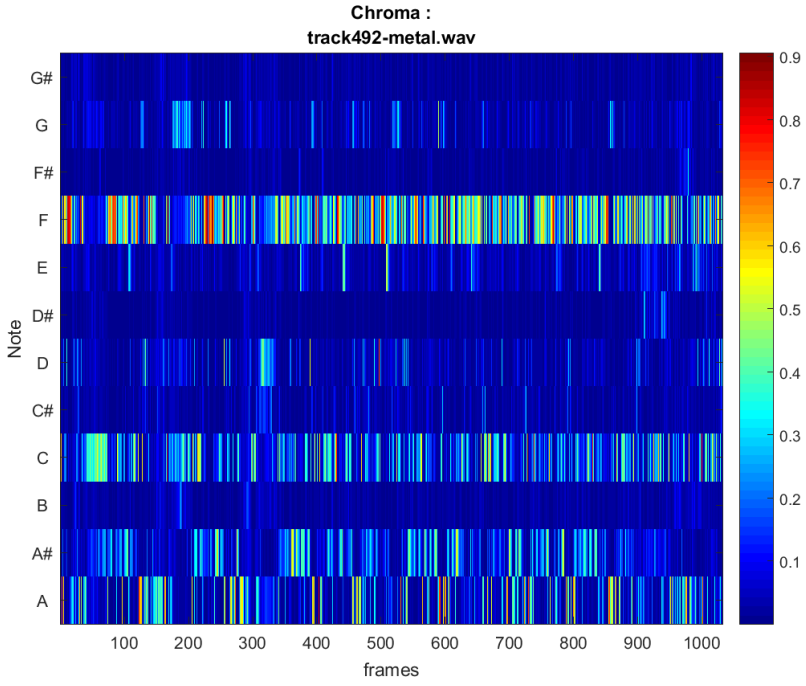

Figure 74: Pitch Class Profile 492

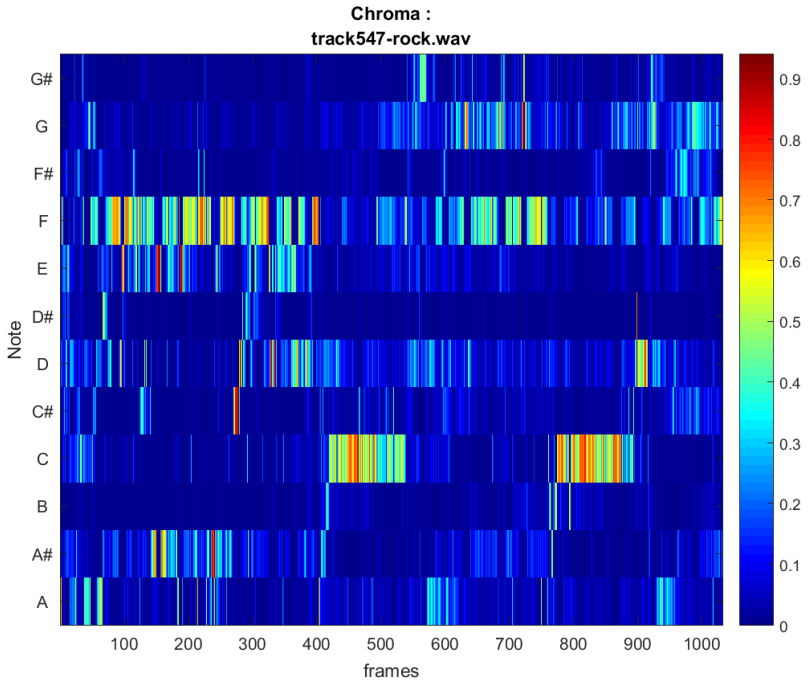

Figure 75: Pitch Class Profile 547

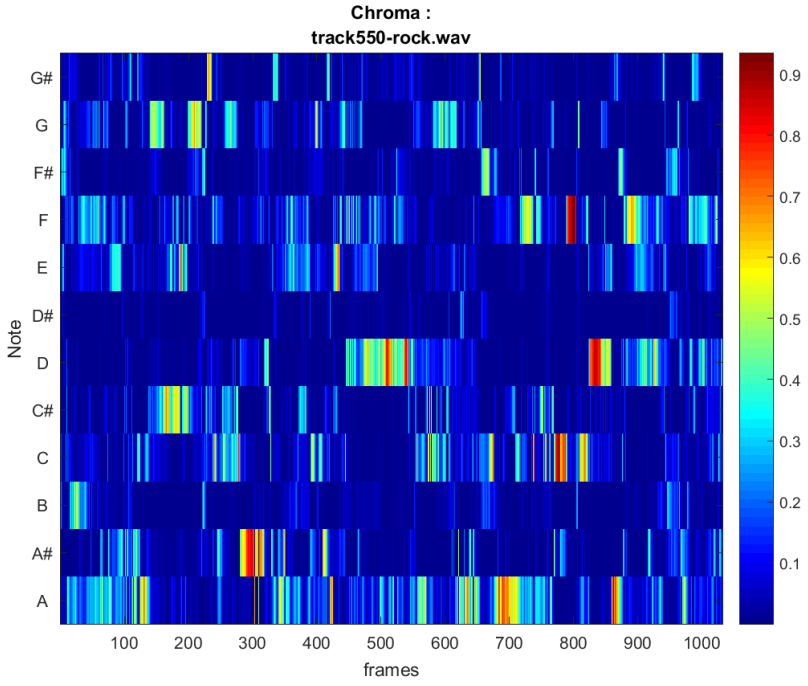

Figure 76: Pitch Class Profile 550

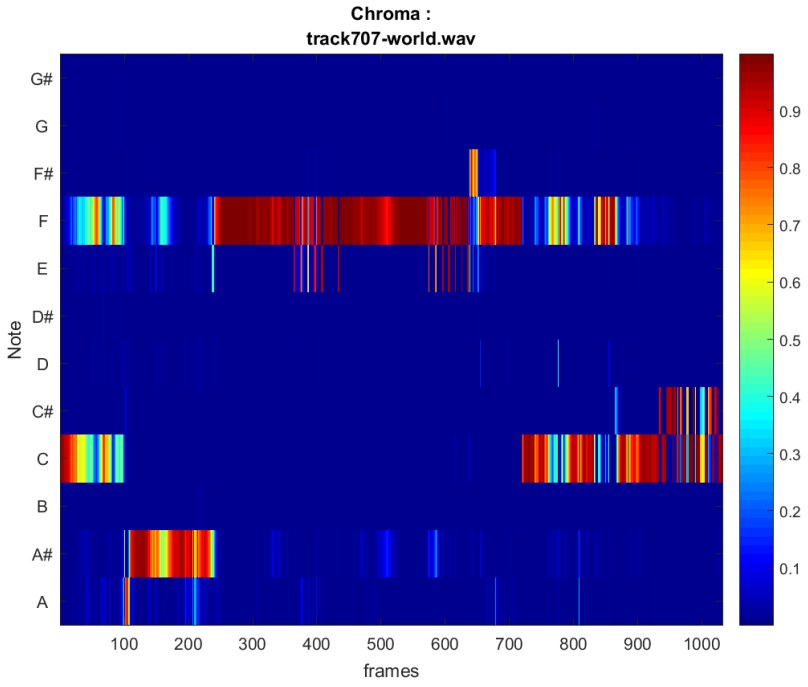

Figure 77: Pitch Class Profile 707

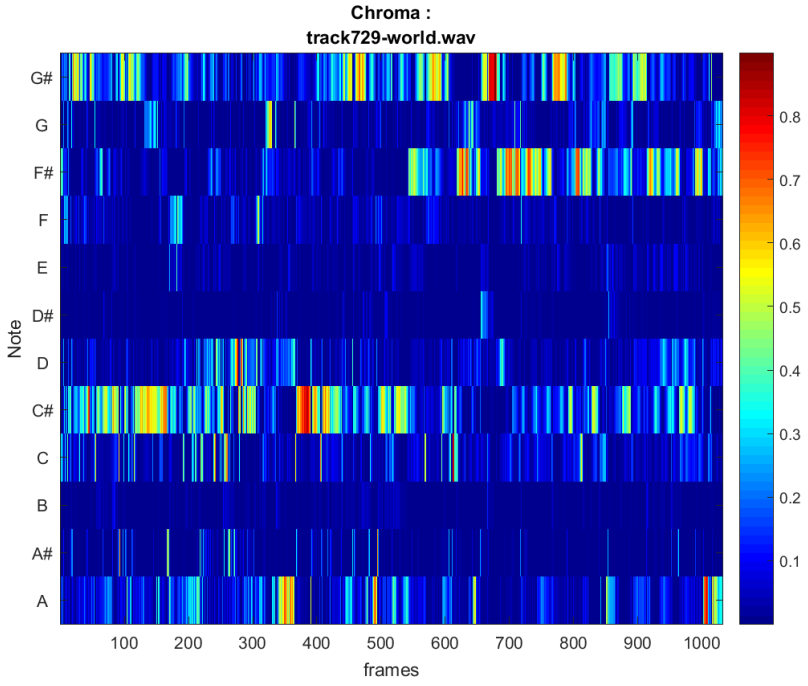

Figure 78: Pitch Class Profile 729

B Code

Listing 6: freqDist.m

```
function [ Xk ] = freqDist(song)
%freqDist Computes the frequency distribution
% info=audioinfo(filename);
% song=extractSound(filename);
frames_overlap = buffer(song,512,256);
w=kaiser(512);
N=512;
Y=fft(w.*frames_overlap(:,2));
Xk=Y(1:256);
K=N/2+1;

for i = 1:length(frames_overlap)
    Y=fft(w.*frames_overlap(:,i));
    Xk(1:K,i)=Y(1:K);
end
end
```

Listing 7: mfcc.m

```
function [ mfccp ] = mfcc( fbank,Xn,~)
%mfcc Computes the Mel frequency coeffecients
%   The mel-spectrum (MFCC) coefficient of the n-th frame is defined for
%   p = 1,...,NB
narginchk(2, 3);
% Xn=freqDist(filename);
mfccp=(fbank*abs(Xn)).^2;
% Lets us still use non-log mfcc (if needed)
if(nargin==3)
    mfccp=10*log10(mfccp);
    return;
end
end
```

Listing 8: spectrumHistogram.m

```
function [ specHistogram ] = spectrumHistogram(filename,mfccp, ~)
%SpectrumHistogram Constructs a 40x50 spectrum histogram
%   Construct a two-dimensional spectrum histogram (a matrix of counts),
%   with 40 columns - one for each mfcc index, and 50 rows - one for each
%   amplitude level, measured in dB, from -20 to 60 dB. You will normalize
%   the histogram, such that the sum along each column (for each "note")
%   is one.
mfccp=10*log10(mfccp);
[a,~]=size(mfccp);
if(a ~= 40)
    errorStruct.message = 'mfccp should have 40 rows';
    errorStruct.identifier = 'SpectrumHistogram:WrongSize';
    error(errorStruct);
end
bottom=min(min(mfccp));
top=max(max(mfccp));
specHistogram = zeros(50,40);
dbSlots = linspace(bottom,top,51);

for i=1:50 %50

    for j=1:a % 40 (numBanks)
        specHistogram(i,j)=numel(find(mfccp(j,:)>dbSlots(i)...
            & mfccp(j,:)<dbSlots(i+1)));
```

```
25
26      end
27  end
28
29  % Normalize function between 0 and 1
30  for i=1:a %40
31      specHistogram(:,i)=specHistogram(:,i)/sum(specHistogram(:,i));
32  end
33
34  % Option to plot and save figure
35  if(nargin == 3)
36      h=figure;
37      imagesc(specHistogram);
38      ax=gca;
39      xlabel('MFCC index');
40      ylabel('Amplitude level (dB)');
41      title({'Spectrum Histogram:'; filename});
42      ax.YTick = flipud(linspace(0,50,11));
43      ax.YTickLabel = fliplr({'-20 dB' '-12 dB' '-4 dB' '4 dB' '12 dB'...
44          '20 dB' '28 dB' '36 dB' '44 dB' '52 dB' '60 dB'});
45      c=colorbar;
46      colormap 'jet';
47      c.Label.String = 'Percent Occurrences of each MFCC coeff.';
48      saveas(gca,['specHistogram' filename(6:end-4) '.png']);
49      close(h);
50  end
51  end
```

Listing 9: simMatrix.m

```
1   function [ sim ] = simMatrix(mfccp,filename,~)
2   %simMatrix displays the similarity matrix for the mel coefficients
3   %   The similarity matrix is defined as follows:
4   %   S(i,j) = sum_{k=1}^{nbanks}{
5   %   (mfcc(k,i)*mfcc(k,j))/(norm(mfcc(:,i)) * norm(mfcc(:,j)))
6   % fbank=melBank();
7   % mfccp=mfcc(fbank,filename);
8   [a,b]=size(mfccp);
9   % Preallocate matrix
10  sim=zeros(b,b);
11  normI=sqrt(sum(abs(mfccp).^2,1));
12  parfor i=1:b %frame i
13      for j=1:b %frame j
14          for k=1:a %fbank k
15              sim(i,j)=sim(i,j)+...
16                  (mfccp(k,i)*mfccp(k,j)/(normI(j).*normI(i)));
17          end
18      end
19  end
20
21  if(nargin == 3)
22      h=figure;
23      imagesc(sim);
24      xlabel('Frame Number');
25      ylabel('Frame Number');
26      title({'Similarity Matrix:'; filename});
27      colorbar;
28      colormap 'jet';
29      saveas(gca,['SimMatrix' filename(6:end-4) '.png']);
30      close(h);
31  end
32  end
```

Listing 10: rhythmIndex.m

```
function [ B ] = rhythmIndex( filename,sim,~)
%rhythmIndex Identifies rhythmic periods
%   B(l) is the sum of all the entries on the lth upper diagonal.
%   The index l associated with the largest value of B(l) corresponds to
%   the presence of a rhythmic period of l*K/f_s seconds
% sim = simMatrix( filename);
[len,~]=size(sim);
B=zeros(1,len);

parfor l=0:len-1
    B(l+1)=(1/(len-l))*sum(diag(sim,l));
end

if(nargin==3)
    h=figure;
    plot(B);
    xlim([0,len]);
    title({'Rhythm Index B(l):'; filename});
    xlabel('lag (frames)');
    ylabel('Presence of rhythmic period');
    saveas(gca,['RhythmIndex' filename(6:end-4) '.png']);
    close(h);
end
end
```

Listing 11: autoC.m

```
function [ AR ] = autoC( filename,sim,~ )
%autoC Computes the autocorrelation of a song
%   In general, if two frames i and j are similar, we can find out
%   if they are repeated later in the segment, at time j+l

info = audioinfo(filename);
% sim = simMatrix( filename);
[len,~]=size(sim);
AR=zeros(1,len);

parfor l=0:len-1
    for i=1:len
        for j=1:len-l
            AR(l+1)=AR(l+1)+(sim(i,j)*sim(i,j+l));
        end
    end
    AR(l+1)=AR(l+1)*(1/(len*(len-l)));
end

if(nargin==3)
    h=figure;
    xAxis=linspace(0,len/info.SampleRate,len);
    plot(xAxis,AR);
    xlim([0,len/info.SampleRate]);
    title({'Autocorrelation AR(l):'; filename});
    xlabel('Lag (secs)');
    ylabel('Autocorrelation');
    saveas(gca,['AutoC' filename(6:end-4) '.png']);
    close(h);
end

end
```

Listing 12: rhythmVar.m

```
function [ ARm ] = rhythmVar( filename,sim,~ )
%autoC Computes the autocorrelation of a song
%   In general, if two frames i and j are similar, we can find out
```

```
%   if they are repeated later in the segment, at time j+l
%   For the sake of this lab, we will be processing 20 frames (~1 second)
% sim = simMatrix( filename);
[len,~]=size(sim);
ARm=zeros(20,floor(len/20));

for m=0:floor(len/20-1)
    Window = sim(20*m+1:20*(m+1),20*m+1:20*(m+1));
    for l=0:19
        temp=Window.*circshift(Window,[0,-l]);
        ARm(l+1,m+1) = sum(sum(temp(1:20-l,:)))/((20-l));
    end
end


ARm=ARm/20;

if(nargin==3)
    h=figure;
    imagesc(ARm);
    ax=gca;
    title({'Autocorrelation AR(l,m):'; filename});
    xlabel('Time (secs)');
    ylabel('Lag (secs)');
    colormap 'jet';
    colorbar;
    saveas(gca,['RhythmVar' filename(6:end-4) '.png']);
    close(h);
end
end
```

Listing 13: normPCP.m

```
function [PCP] = normPCP( filename,x,~)
%rhythmIndex Identifies rhythmic periods
%   B(l) is the sum of all the entries on the lth upper diagonal.
%   The index l associated with the largest value of B(l) corresponds to
%   the presence of a rhythmic period of l*K/f_s seconds
info=audioinfo(filename);
% x=extractSound(filename);
[s,~,~]=spectrogram(x,kaiser(512),256,512,info.SampleRate,'yaxis');
s=abs(s);
[a,b]=size(s);

% pks=zeros(a,b);
locs=zeros(a,b);
peaks=zeros(a,b);
for i=1:b
    len=length(findpeaks(s(:,i)));
    [peaks(1:len,i),locs(1:len,i)]=findpeaks(s(:,i));
end
freqVals=(5512/257)*locs;
f0=27.5;
sm=round(12*log2(freqVals/f0));
r=12*log2(freqVals/f0)-sm;
c=mod(sm,12);
[~,B]=size(c); % 257,1032

w=(cos(pi*r/2)).^2;
% Make pretty with matrix
w(isnan(w))=0;

% k is the number of peaks
% c is the chroma (A,A#,B,etc.)
% n is the frame number
% Weighting function, w, needs to be k*n*c large

```

```
PCP=zeros(12,B);


% Pick out semitones
% Loop through each value of c (1,2,...,12)
% Find the corresponding rows to extract from w where semitone is
% (1,2,...,12) multiply by Xn at that row at the corresponding row



peaks=peaks.^2;

for i=1:B %nFrames
    for j=0:11
        a=find(c(:,i)==j);
        PCP(12-j,i)=sum(w(a,i).*peaks(a,i));
    end
end
PCP=bsxfun(@rdivide,PCP,sum(PCP));



if(nargin==3)
    h=figure;
    imagesc(PCP);
    ax=gca;
    ax.YTickLabel=fliplr({'A ','A#','B ','C ','C#','D ','D#','E ',...
        'F ','F#','G ','G#'});
    ax.YTick=linspace(1,12,12);
    colormap 'jet';
    title({'Chroma :'; filename});
    xlabel('frames');
    ylabel('Note');
    colorbar;
    saveas(gca,['NPCP' filename(6:end-4) '.png']);
    close(h);
end
end
```

Listing 14: buildReport.m

```
%build everything necessary for report
clear filename song Xk mfccp specHistogram sim ARm NPCP;
close all;
fileArray = cellstr([
    'track201-classical.wav '; ...
    'track204-classical.wav '; ...
    'track370-electronic.wav'; ...
    'track396-electronic.wav'; ...
    'track437-jazz.wav      '; ...
    'track439-jazz.wav      '; ...
    'track463-metal.wav     '; ...
    'track492-metal.wav     '; ...
    'track547-rock.wav      '; ...
    'track550-rock.wav      '; ...
    'track707-world.wav     '; ...
    'track729-world.wav     '
    ]);
fbank=melBank();
parfor i=1:12
    filename=['' fileArray{i} ''];
    song=extractSound(filename);
    Xk=freqDist(song);
    mfccp=mfcc(fbank,Xk);
    specHistogram=spectrumHistogram(filename,mfccp,1);
    sim=simMatrix(mfccp,filename,1);
    rhythmIndex(filename,sim,1);
```

```
27      autoC(filename,sim,1);
28      ARm=rhythmVar( filename,sim,1);
29      NPCP=normPCP(filename,song,1);
30  end
```