Lab 2

Clint Olsen

ECEN 4532: DSP Lab

February 20, 2018

University of Colorado
Boulder

Contents

Introduction **2**

Assignment 1 **2**

Assignment 2 **2**

Assignment 3 **2**

Assignment 4 **5**

Assignment 5 **8**

Assignment 6 **11**

Assignment 7 **11**

Assignment 8 **11**

Assignment 9 **14**

Assignment 10 **17**

Assignment 11 **19**

Code Appendix **23**

Introduction

This lab is a continuation of the first lab in this course, which focused on low-level characteristics of audio analysis using MATLAB. In this lab, analysis will be continued on the same audio signals, but at a higher level of analysis. Aspects such as rhythm and tonality will be characterized for various genres and will be mathematically and graphically represented to show the correlation between the data analysis and tangible aural properties. This will be the beginnings of obtaining information about music that can lead to a better means of categorizing and organizing individual tracks autonomously.

Assignment 1

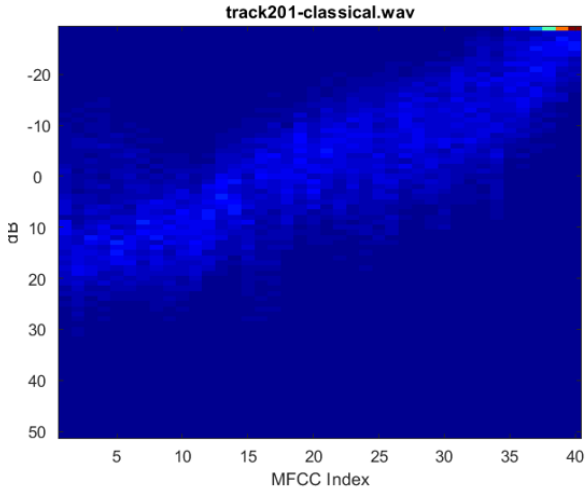
This assignment is just an extension of the last part of the previous lab, where the MFCC coefficients were calculated for describing the power distribution of the frequency content of a given signal after being passed through the 40 bank cochlear filter bank. Extra computation is performed here to convert the values of these coefficients to the dB scale. See Assignment 1 Appendix for MATLAB code.

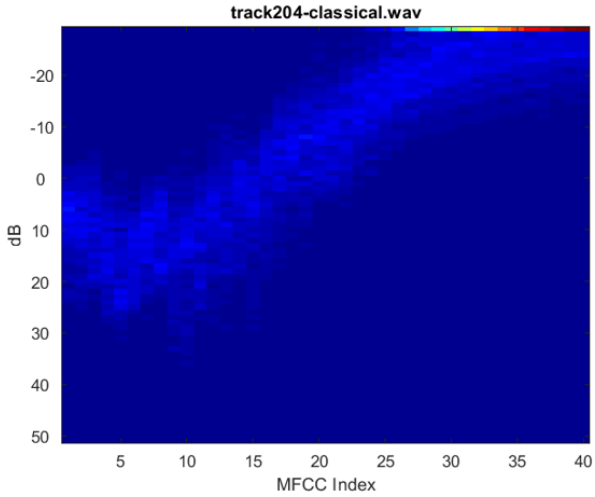
Assignment 2

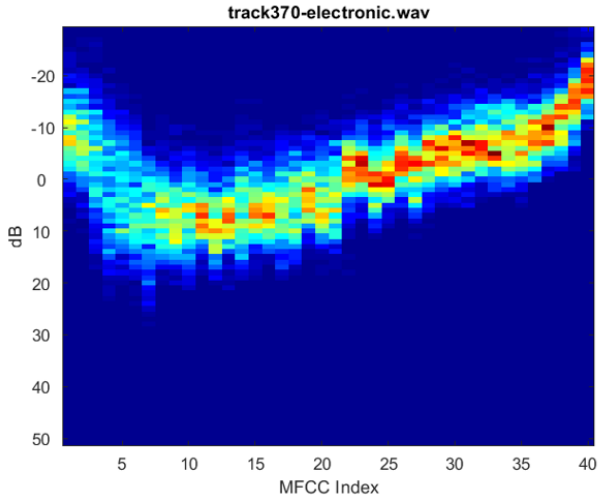
In order to visualize the MFCC coefficients, this portion of the lab focuses on creating a $81x40$ matrix of counts to characterize the intensity of the frequency content at different power levels for each frequency response of the cochlear filter bank. This is done by summing the number of times a given power level is achieved at a particular filter. The dimension of the matrix come from the range of the dB scale used (-20 to 60 dB) resulting in 81 rows in the matrix, and the 40 columns from the 40 MFCC coefficients for each filter banks used in the cochlear filter bank. The MATLAB code for the development of this matrix is shown in the Assignment 2 Appendix.

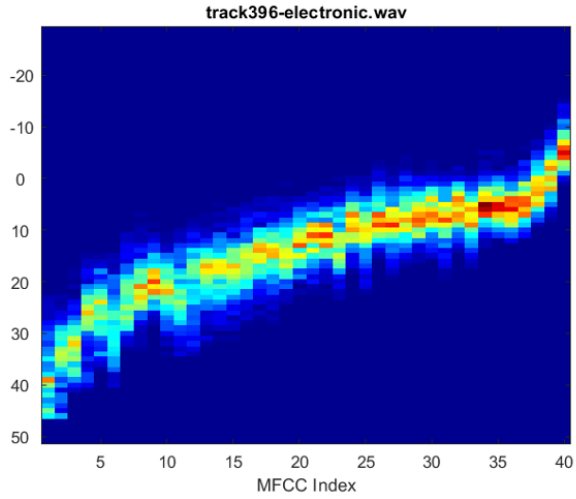
Assignment 3

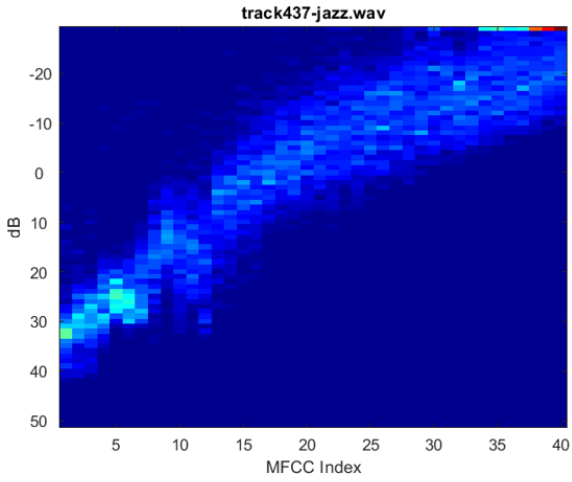
Below are the plots associated with the spectral histogram for each of the 12 tracks:

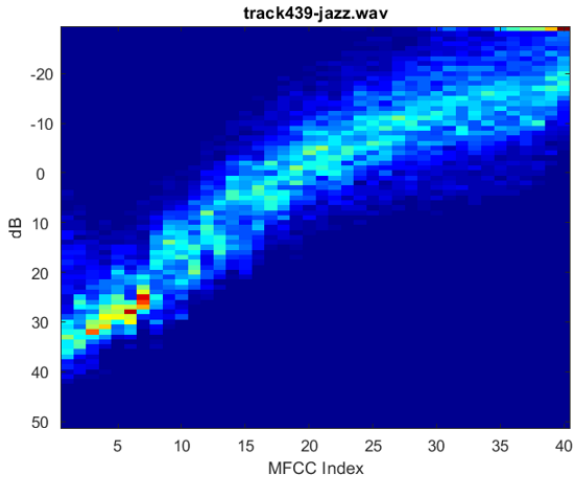

track201-classical.wav
-20
-10
0
10
20
30
40
50
dB
5
10
15
20
25
30
35
40
MFCC Index

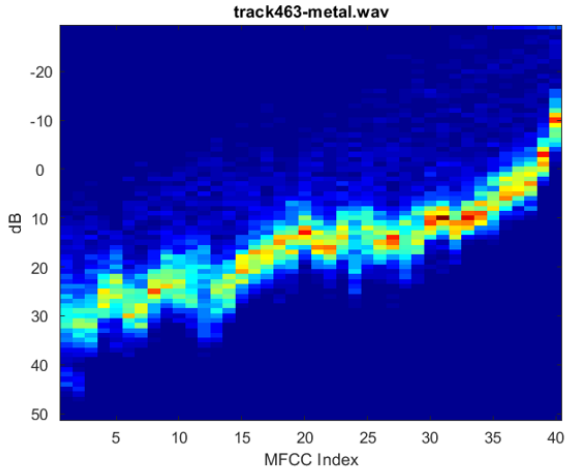

track204-classical.wav
-20
-10
0
10
20
30
40
50
dB
5
10
15
20
25
30
35
40
MFCC Index

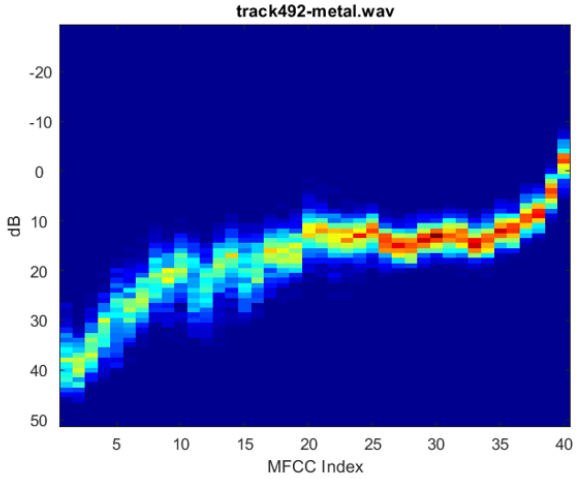

track370-electronic.wav
-20
-10
0
10
20
30
40
50
dB
5
10
15
20
25
30
35
40
MFCC Index

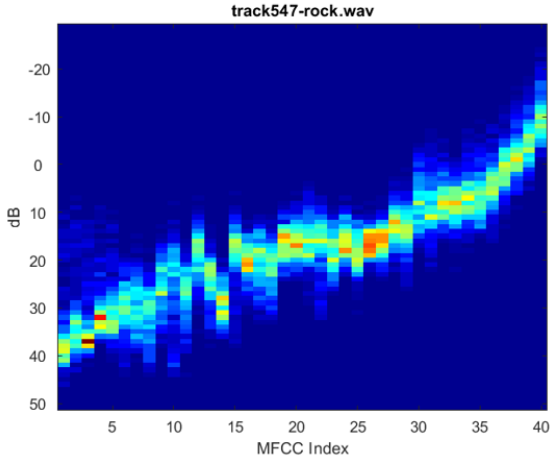

track396-electronic.wav
-20
-10
0
10
20
30
40
50
dB
5
10
15
20
25
30
35
40
MFCC Index

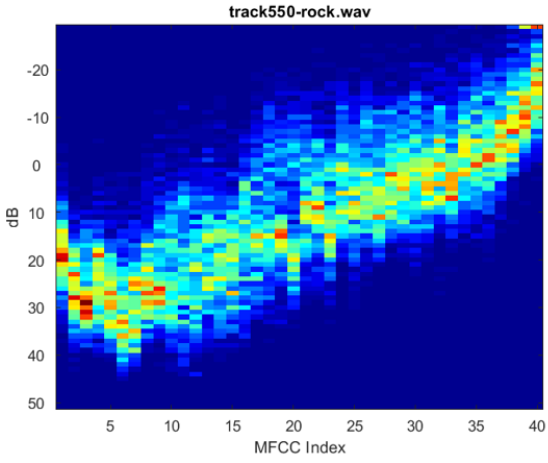

track437-jazz.wav
-20
-10
0
10
20
30
40
50
dB
5
10
15
20
25
30
35
40
MFCC Index

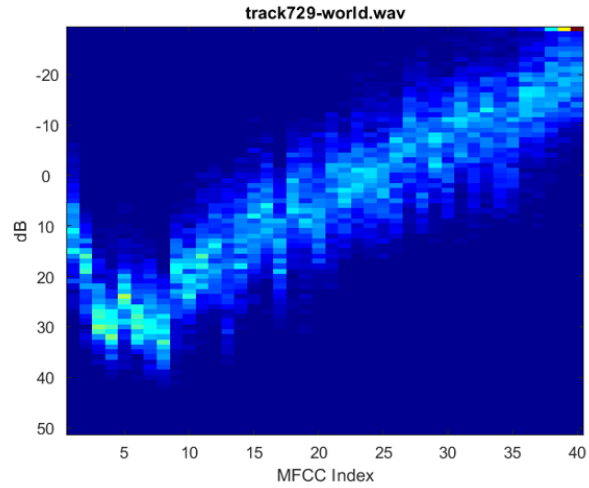

track439-jazz.wav
-20
-10
0
10
20
30
40
50
dB
5
10
15
20
25
30
35
40
MFCC Index

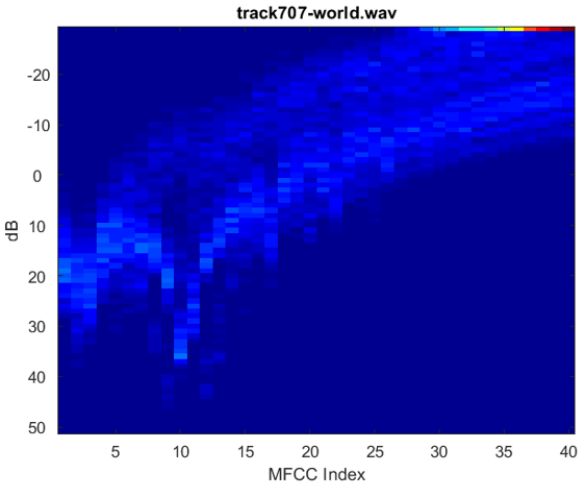

track463-metal.wav
dB
-20
-10
0
10
20
30
40
50
5
10
15
20
25
30
35
40
MFCC Index


track492-metal.wav
dB
-20
-10
0
10
20
30
40
50
5
10
15
20
25
30
35
40
MFCC Index


track547-rock.wav
dB
-20
-10
0
10
20
30
40
50
5
10
15
20
25
30
35
40
MFCC Index


track550-rock.wav
dB
-20
-10
0
10
20
30
40
50
5
10
15
20
25
30
35
40
MFCC Index

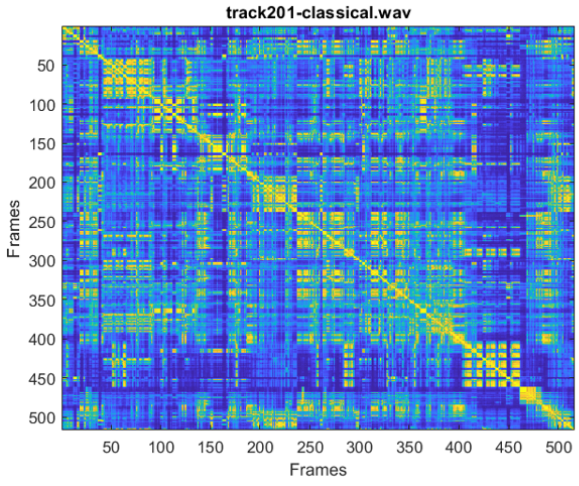

track729-world.wav
dB
-20
-10
0
10
20
30
40
50
5
10
15
20
25
30
35
40
MFCC Index

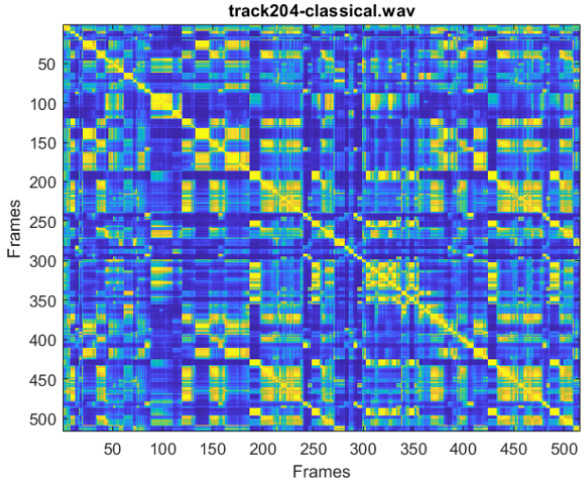

track707-world.wav
dB
-20
-10
0
10
20
30
40
50
5
10
15
20
25
30
35
40
MFCC Index

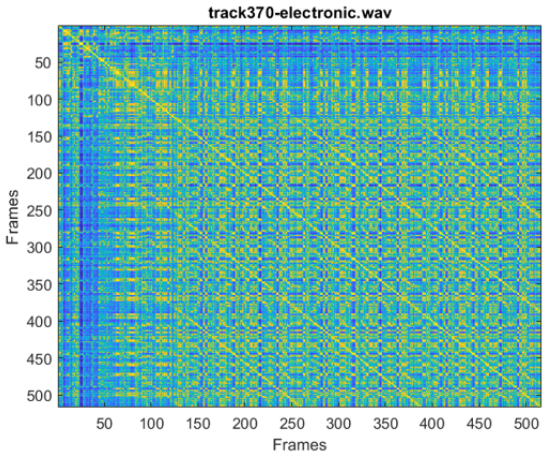
Many interesting things can be seen from the above plots. Each genre has a unique character to it in terms of the spectral intensity that correlates well to what is actually heard. For example, the metal tracks have many high intensity MFCC coefficients at a high decibel levels in comparison to a track such as jazz 437, which has some high count of low frequency components at high decibel values, but overall lacks the constant high power across frequencies that the metal tracks have. These graphs can give an overall impression of the dynamic quality of a track and what frequencies these dynamics occur, based on the MFCC coefficients.

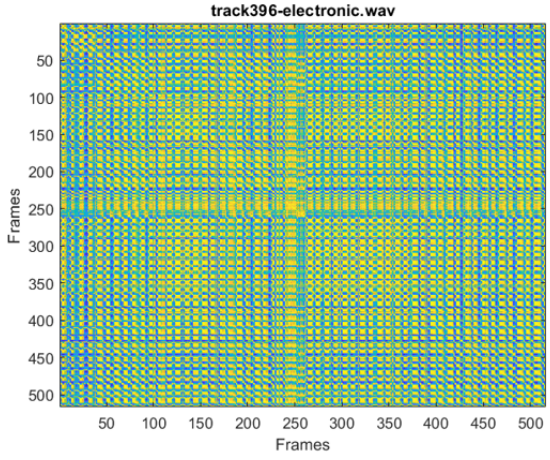
Assignment 4

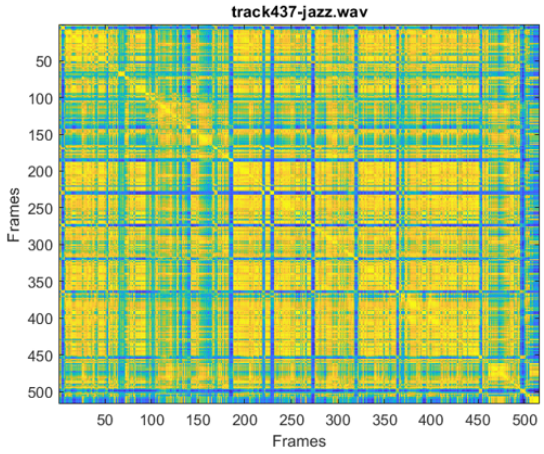
One way to begin building a sense of rhythm mathematically is to look at the the similarity of vectors (in this case frames) within a given audio track. This can be done using the dot product which is a mathematical method to determine how "similar" two given vectors are. To start, each frame in an audio track can be dotted with each other to build a basic characterization of how similar frames are. This is shown for each track through the use of a similarity matrix constructed by the MATLAB code in the Assignment 4 Appendix:

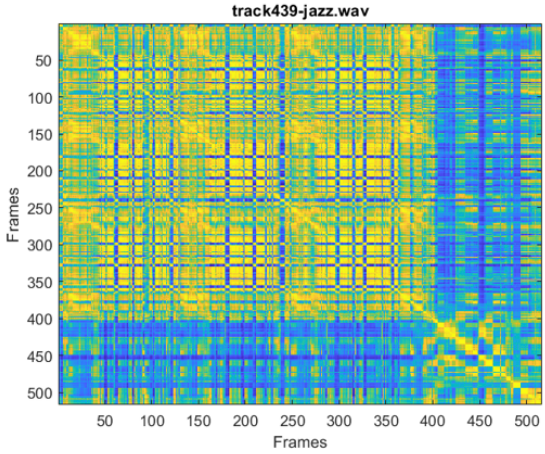


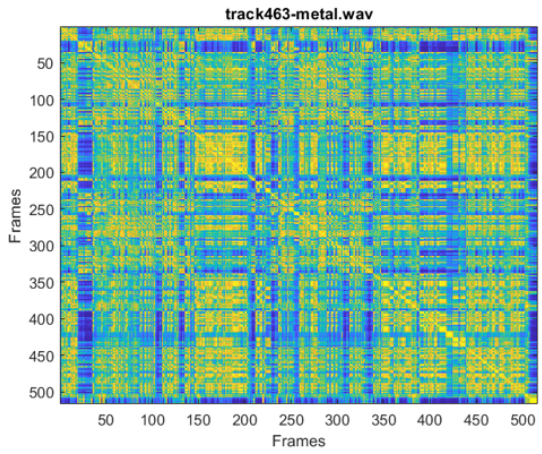


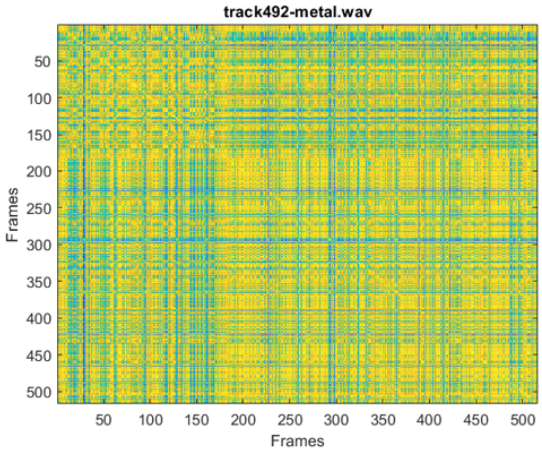

track370-electronic.wav
Frames
Frames
50 100 150 200 250 300 350 400 450 500

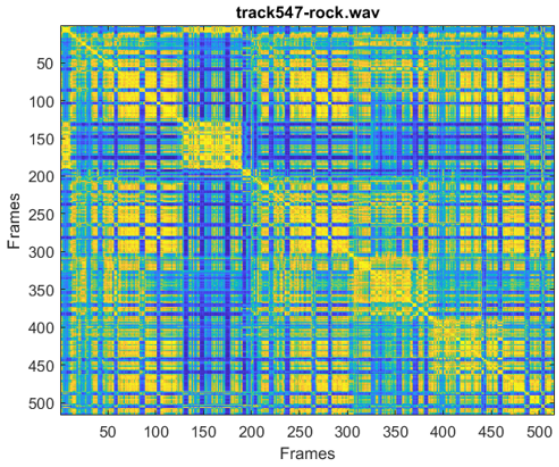

track396-electronic.wav
Frames
Frames
50 100 150 200 250 300 350 400 450 500

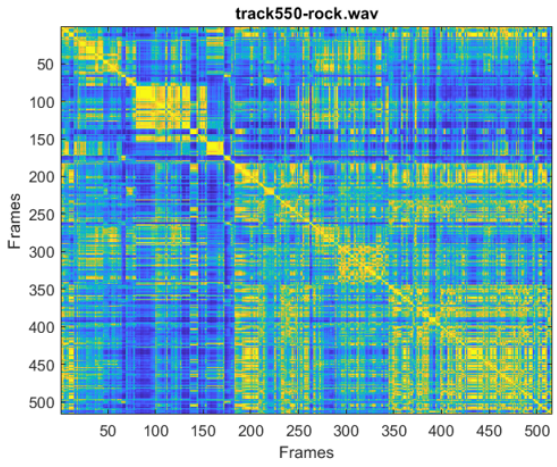

track437-jazz.wav
Frames
Frames
50 100 150 200 250 300 350 400 450 500

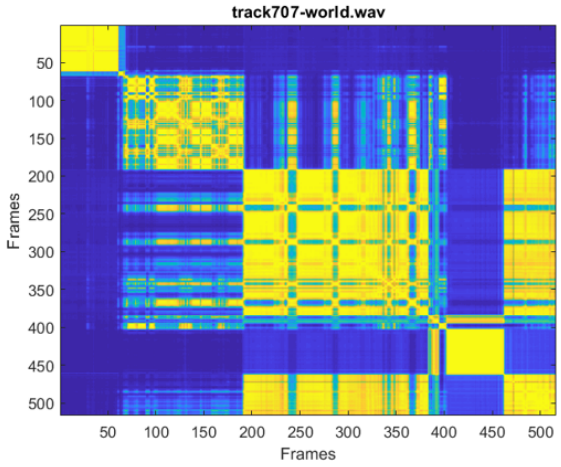

track439-jazz.wav
Frames
Frames
50 100 150 200 250 300 350 400 450 500

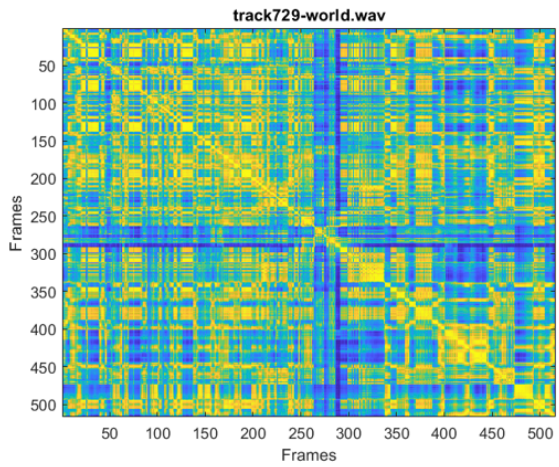

track463-metal.wav
Frames
Frames
50 100 150 200 250 300 350 400 450 500


track492-metal.wav
Frames
Frames
50 100 150 200 250 300 350 400 450 500


track547-rock.wav
Frames
Frames
50
100
150
200
250
300
350
400
450
500


track550-rock.wav
Frames
Frames
50
100
150
200
250
300
350
400
450
500

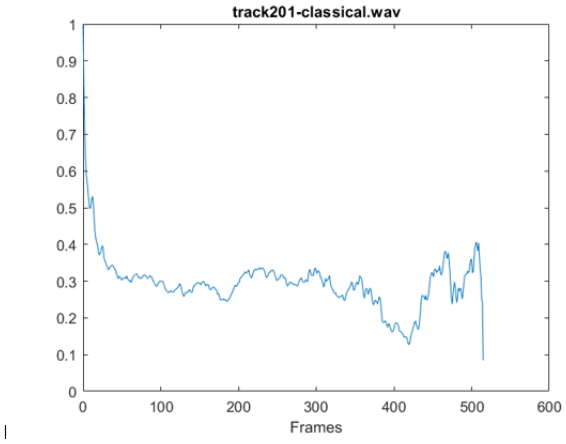

track707-world.wav
Frames
Frames
50
100
150
200
250
300
350
400
450
500

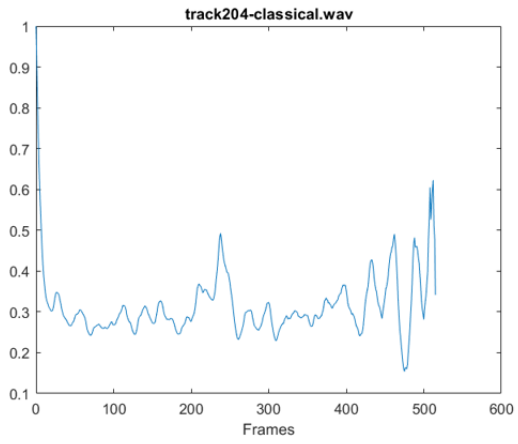

track729-world.wav
Frames
Frames
50
100
150
200
250
300
350
400
450
500

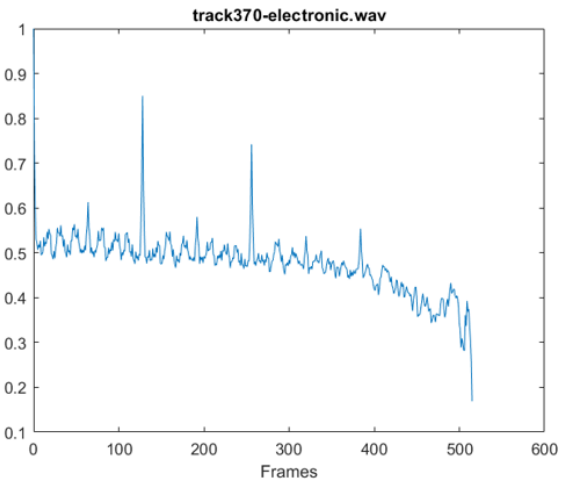
These interesting looking plots can give a lot of information regarding the audio characteristics of a audio track. Each index of the matrix holds the value of how "related" the frame at i is to a frame at j. The magnitude of this value is the dot product of the two frame vectors. Two interesting examples of this can show rhythmic consistency and sustained notes. In the electronic 396, it can be seen that the yellow colors are evenly separated, showing that most frames are similar to those of a particular number of frames away from them. Upon listening to this track, the electronic bass drum is very constant, so this even spacing gives a sense of the steady rhythm produced by the bass drum. In looking at the world 707 track, there are blocks of frames next to each other that share commonalities for long periods of time. This can be attributed to sustained notes over many frames that are evident when listening to track.

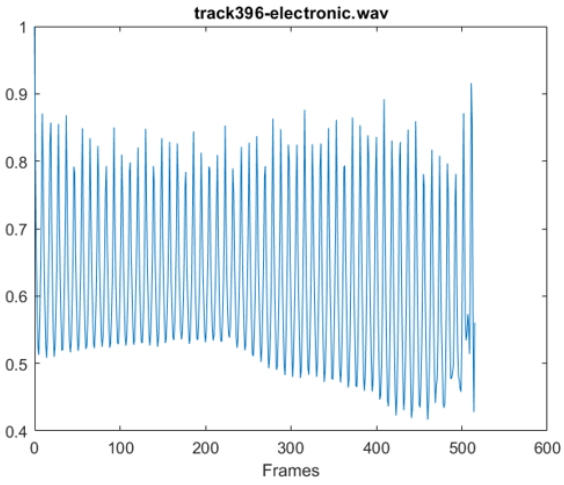
Assignment 5

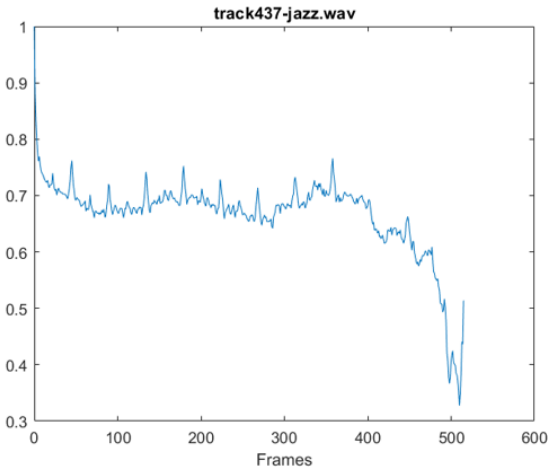
As referenced in the last portion of the lab, the similarity matrix can give a sense of rhythm, if there is a consistent correlation between frames that are a certain distance apart. This is called lag between the frames, which can be used to begin determining the tempos within a track. This is done by summing all of the elements above the main diagonal in the similarity matrix, normalizing, and seeing for which lag value of l in the summation yields the highest result (See Assignment 5 Appendix for MATLAB code). This tells us the period between rhythmic repetition. Below is the results of this computation for the 12 tracks:

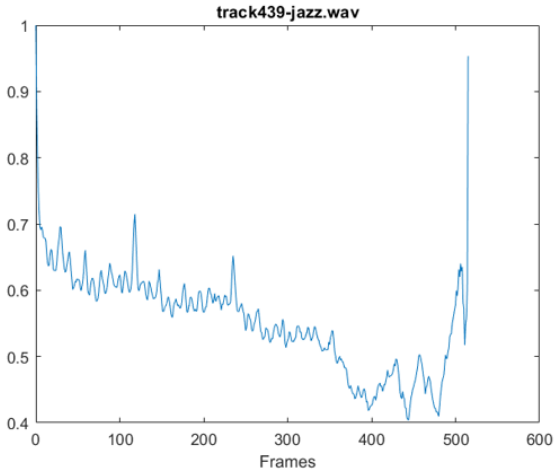


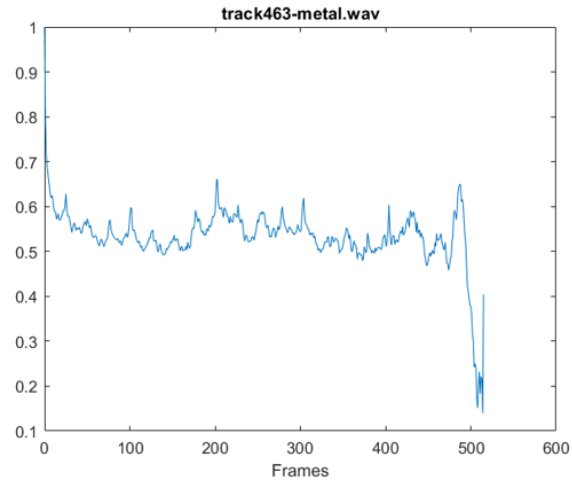


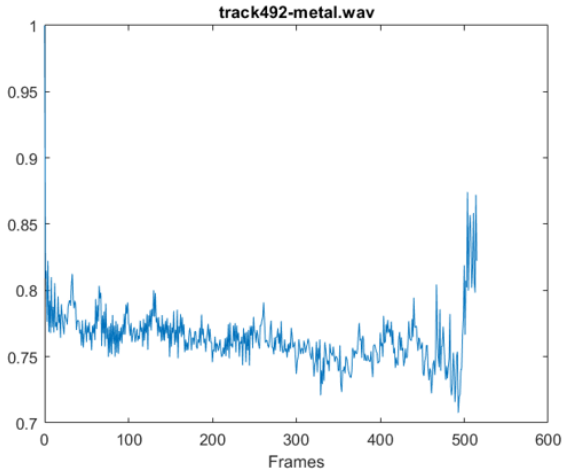

track370-electronic.wav
Frames

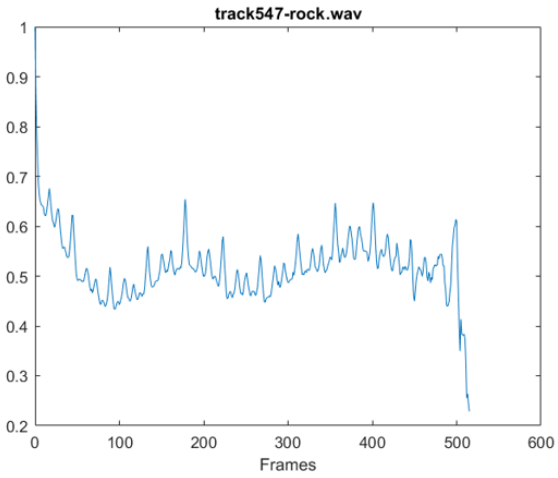

track396-electronic.wav
Frames

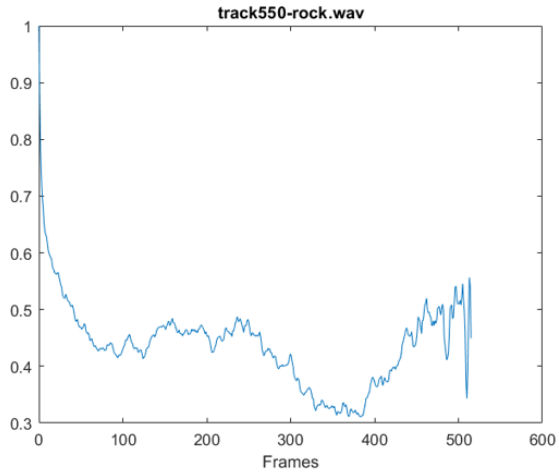

track437-jazz.wav
Frames

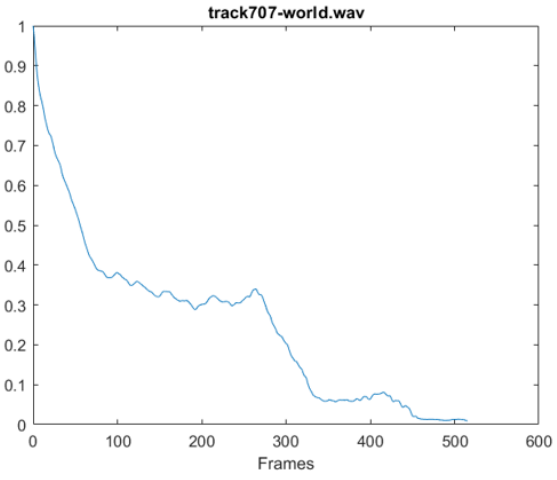

track439-jazz.wav
Frames

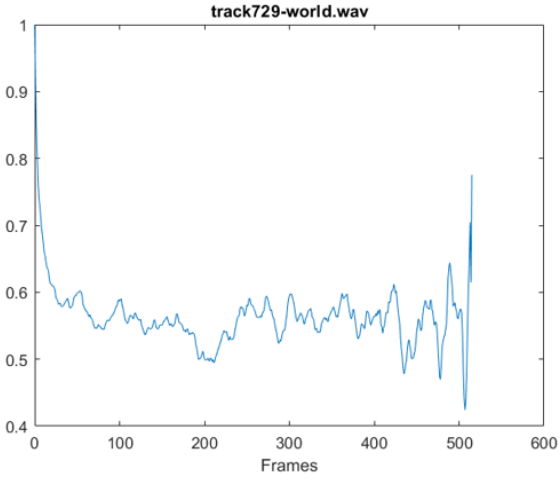

track463-metal.wav
Frames


track492-metal.wav
Frames


track547-rock.wav
Frames


track550-rock.wav
Frames


track707-world.wav
Frames


track729-world.wav
Frames

The best example to look at to understand these plots is the electronic 396 track. It can be seen that there are many evenly spaced spikes that correlate to high similarity to the number of frames away where that spike exists on the x axis. In this example, each spike is spaced multiples of a particular lag, which determines the tempo. However, looking at the classical and world tracks, the lack of consistent spikes indicates little presence of a set tempo, which is true in regards to the more free form nature of the genres. Also, rhythmic patterns cannot be seen because these genres do not have a strong tempo driving force like the bass drum in the electronic tracks.

Assignment 6

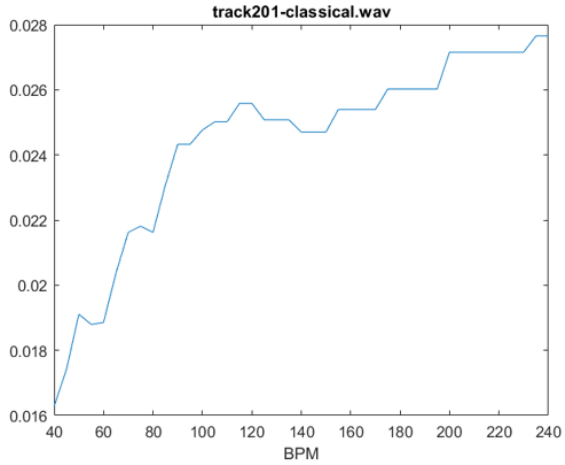
In order to determine the true tempo of a track in BPM, we can again use the similarity matrix. First, the median of the matrix must be determined for use in computing the lag value associated with a given tempo in beats per minute (BPM). The MATLAB code for this can be found in the Assignment 6 Appendix.

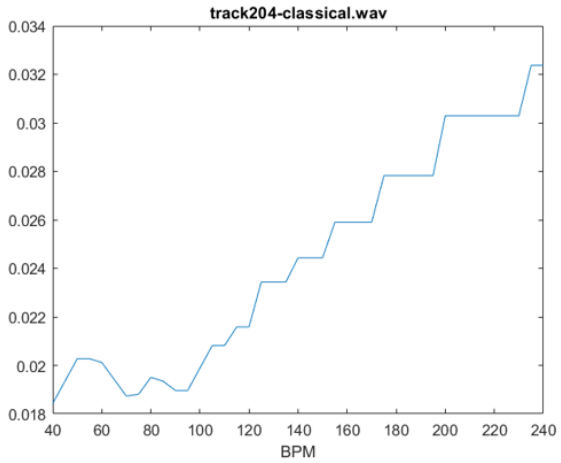
Assignment 7

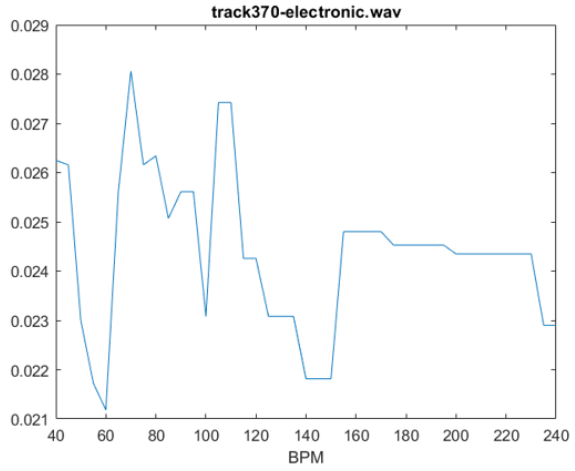
The next step in determining the BPM is to calculate the number of times that the similarity matrix is larger than the median of the matrix at various lag values correlated with BPM values from 40 to 240 in steps of 5 BPM. This will result in large counts for highly similar lags that are common throughout the track. The code for this is shown the Assignment 7 Appendix.

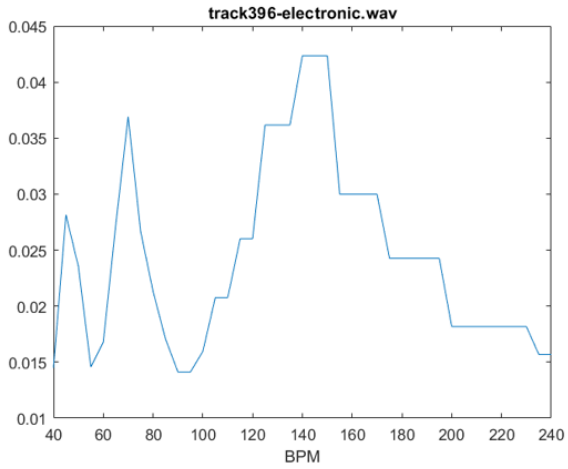
Assignment 8

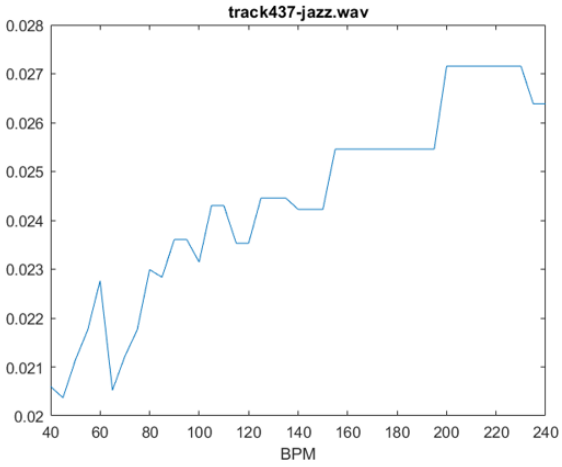
Finally, a histogram of the BPM values determined for a track can be plotted for each of the 12 track:

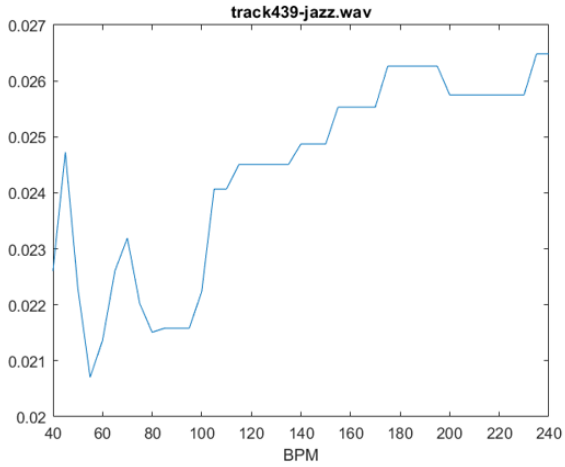


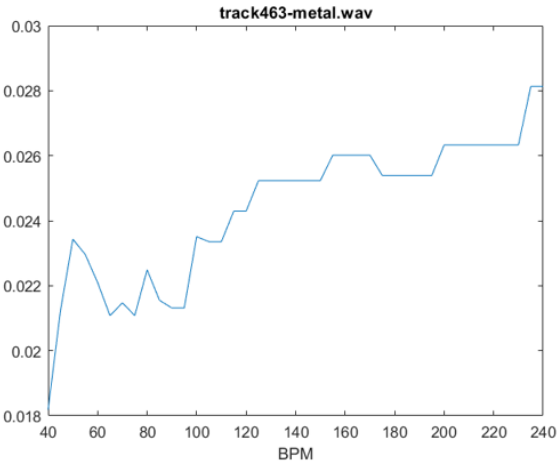


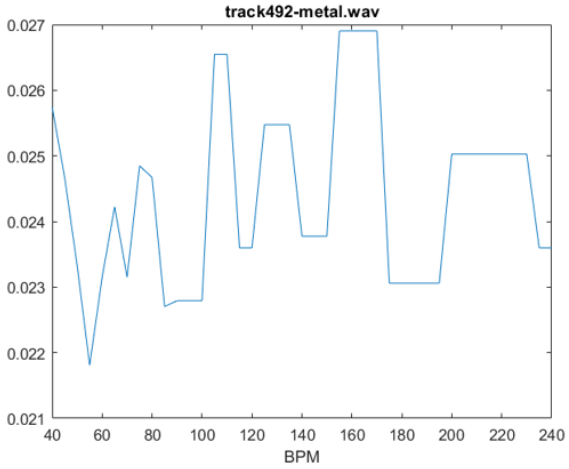

track370-electronic.wav
0.029
0.028
0.027
0.026
0.025
0.024
0.023
0.022
0.021
40 60 80 100 120 140 160 180 200 220 240
BPM

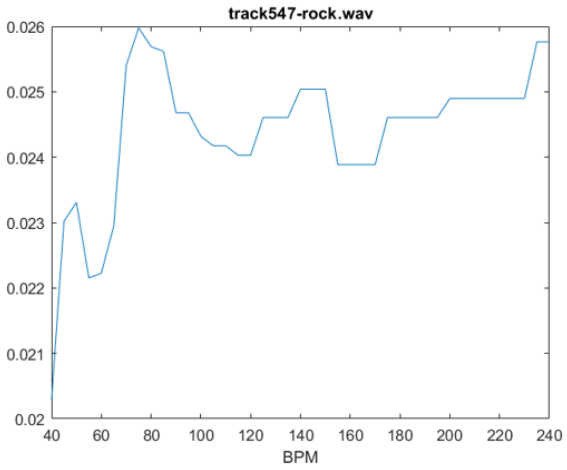

track396-electronic.wav
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
40 60 80 100 120 140 160 180 200 220 240
BPM

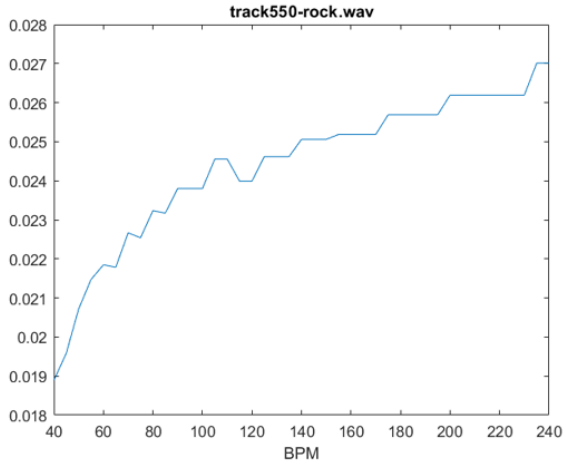

track437-jazz.wav
0.028
0.027
0.026
0.025
0.024
0.023
0.022
0.021
0.02
40 60 80 100 120 140 160 180 200 220 240
BPM

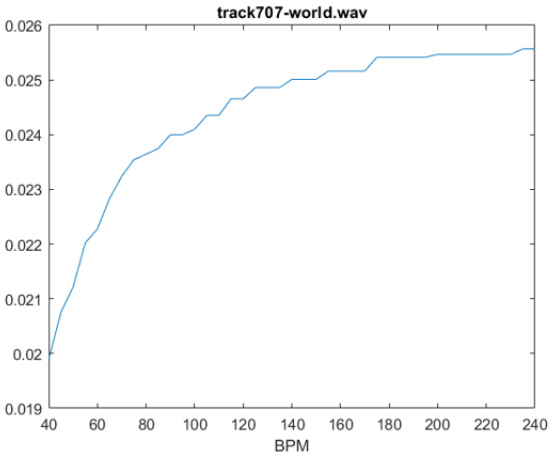

track439-jazz.wav
0.027
0.026
0.025
0.024
0.023
0.022
0.021
0.02
40 60 80 100 120 140 160 180 200 220 240
BPM

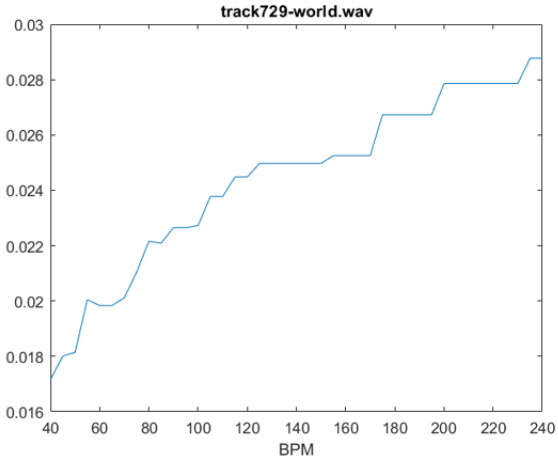

track463-metal.wav
0.03
0.028
0.026
0.024
0.022
0.02
0.018
40 60 80 100 120 140 160 180 200 220 240
BPM


track492-metal.wav
0.027
0.026
0.025
0.024
0.023
0.022
0.021
40 60 80 100 120 140 160 180 200 220 240
BPM


track547-rock.wav
0.026
0.025
0.024
0.023
0.022
0.021
0.02
40 60 80 100 120 140 160 180 200 220 240
BPM


track550-rock.wav
0.028
0.027
0.026
0.025
0.024
0.023
0.022
0.021
0.02
0.019
0.018
40 60 80 100 120 140 160 180 200 220 240
BPM

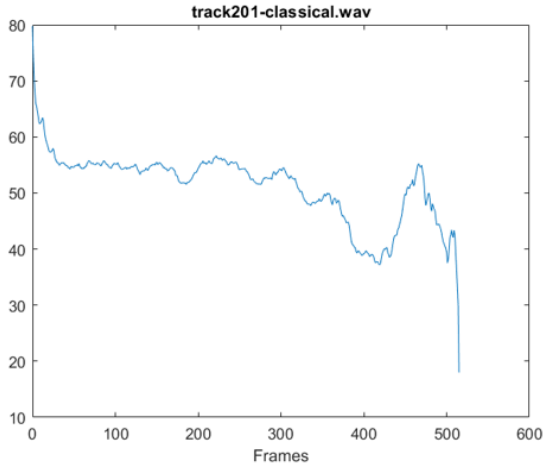

track707-world.wav
0.026
0.025
0.024
0.023
0.022
0.021
0.02
0.019
40 60 80 100 120 140 160 180 200 220 240
BPM

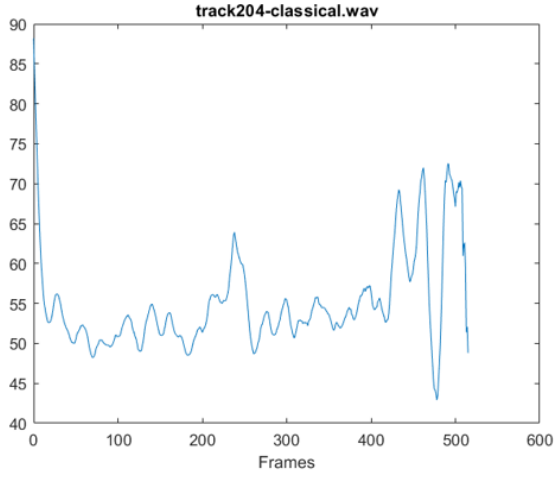

track729-world.wav
0.03
0.028
0.026
0.024
0.022
0.02
0.018
0.016
40 60 80 100 120 140 160 180 200 220 240
BPM

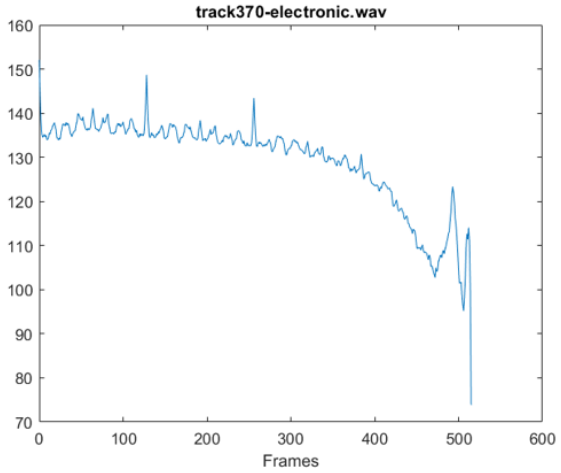
Once again, the electronic 396 track gives a great example of rhythmic consistency to understand what is occurring in the graph. A large spike can be seen right around 145 BPM, which means that the similarity matrix held many values greater than its median at the lag value that correlated to 145 BPM. Interestingly enough, I took the original track and tapped the tempo into a metronome tool, which showed the actual tempo of the song is 140 BPM, so this is a good approximation for the tempo of a given song. This is only true, however, if there is a strong rhythmic force driving the song, like the bass drum in electronic 396. The World 707 track is almost a perfectly smooth curve, which indicates there is nothing interesting that can be determined about the tempo, because in this case, the world track is more Rubato in tempo.

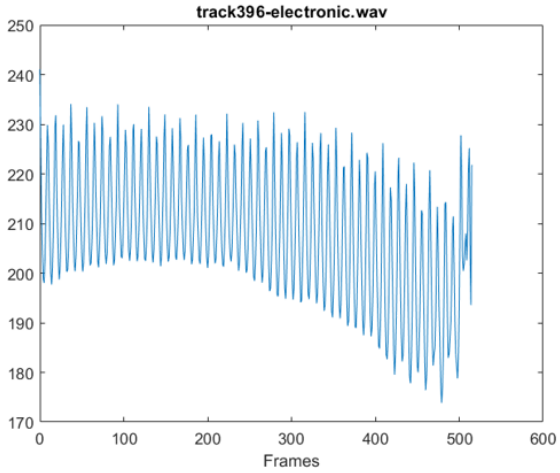
Assignment 9

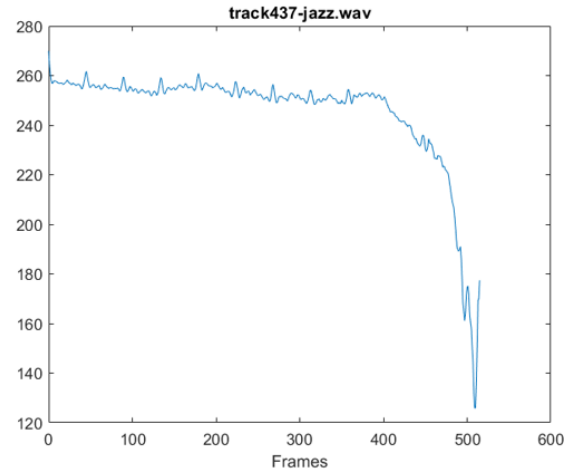
In order to get an even better approximation for the rhythmic patterns within a track, the autocorrelation can be used instead of the previous algorithm in Assignment 5. The difference with this approach is that the correlation is found through taking the dot product of the similarity matrix with itself at different lag values to determine the lag that correlates with a particular BPM. This is a better approximation due to the need for a reference to compare the result with. The MATLAB code for this is shown in the Assignment 9 Appendix and the plots for the autocorrelation of the 12 tracks is shown below:

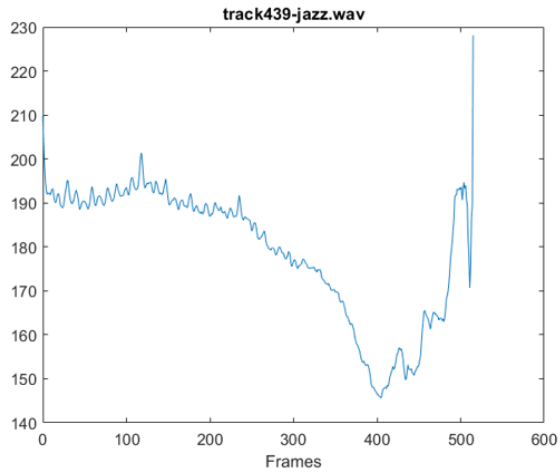


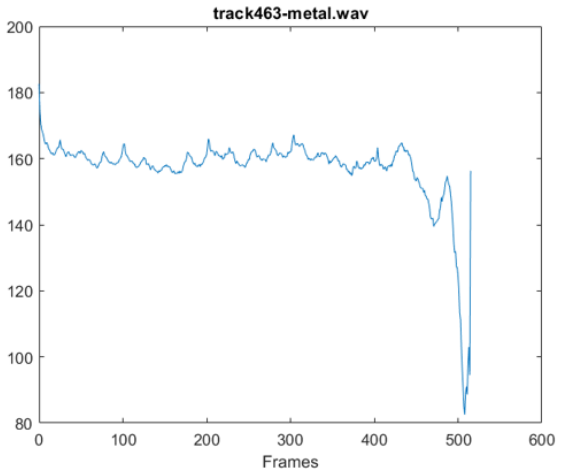


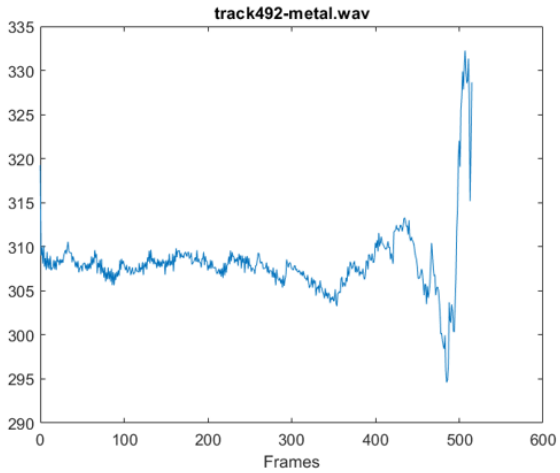

track370-electronic.wav
160
150
140
130
120
110
100
90
80
70
0
100
200
300
400
500
600
Frames

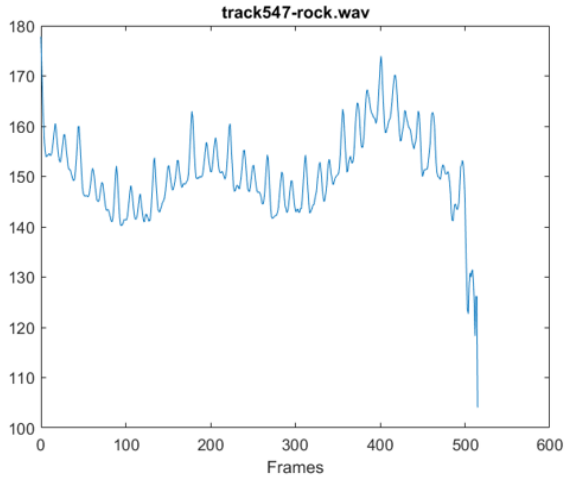

track396-electronic.wav
250
240
230
220
210
200
190
180
170
0
100
200
300
400
500
600
Frames

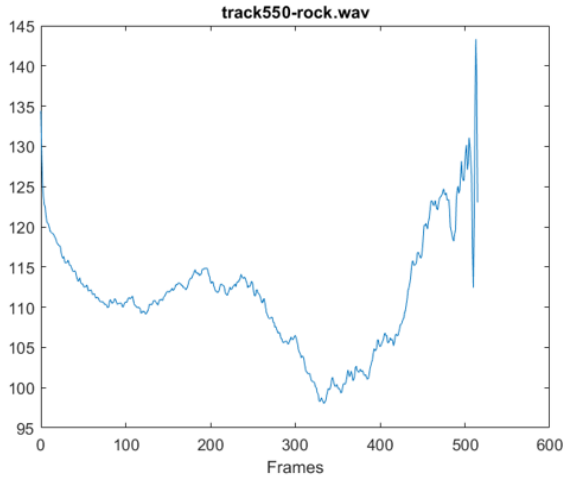

track437-jazz.wav
280
260
240
220
200
180
160
140
120
0
100
200
300
400
500
600
Frames

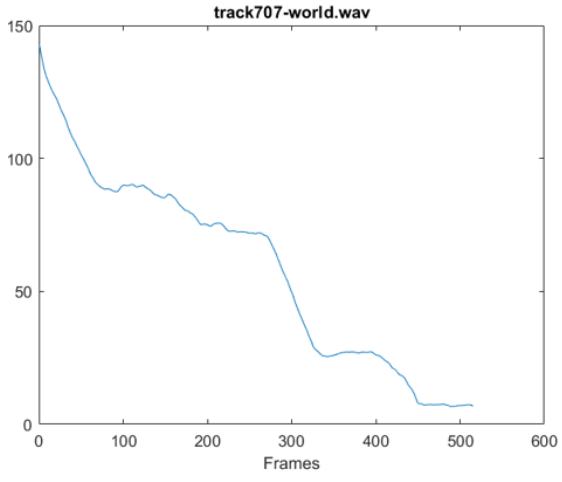

track439-jazz.wav
230
220
210
200
190
180
170
160
150
140
0
100
200
300
400
500
600
Frames

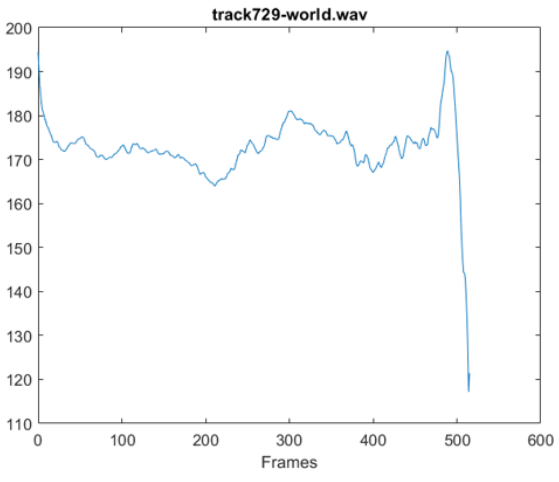

track463-metal.wav
200
180
160
140
120
100
80
0
100
200
300
400
500
600
Frames


track492-metal.wav
335
330
325
320
315
310
305
300
295
290
0
100
200
300
400
500
600
Frames


track547-rock.wav
180
170
160
150
140
130
120
110
100
0
100
200
300
400
500
600
Frames


track550-rock.wav
145
140
135
130
125
120
115
110
105
100
95
0
100
200
300
400
500
600
Frames

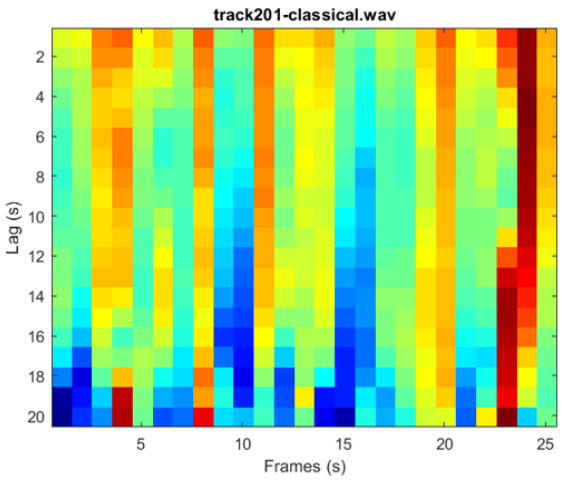

track707-world.wav
150
100
50
0
0
100
200
300
400
500
600
Frames

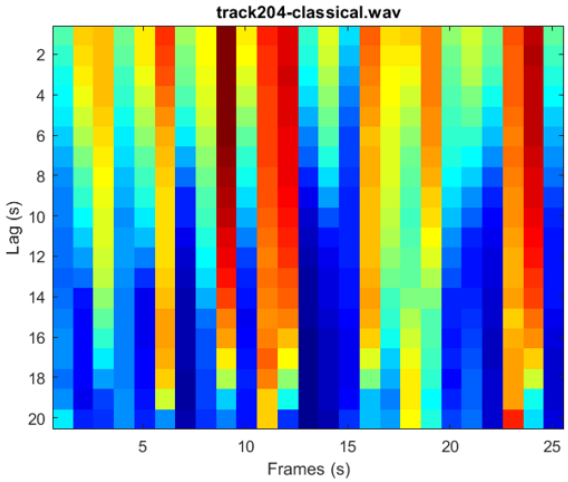

track729-world.wav
200
190
180
170
160
150
140
130
120
110
0
100
200
300
400
500
600
Frames

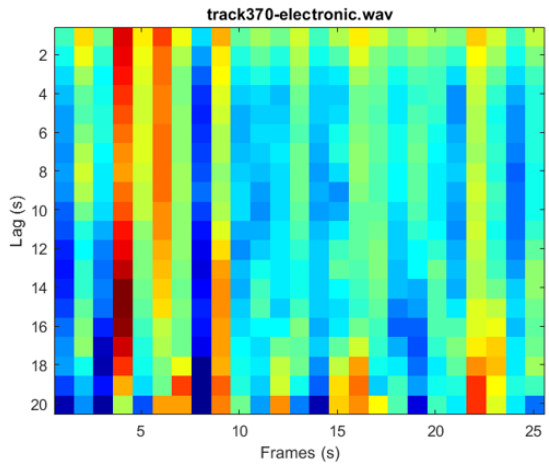
The results of this computation are similar to the first rhythmic calculation where tracks such as electronic give a strong rhythmic pattern, but the world and classical tracks do not. The graphs are also just a bit more refined and if this data was used to determine the BPM, there would be less error involved with resulting BPM value.

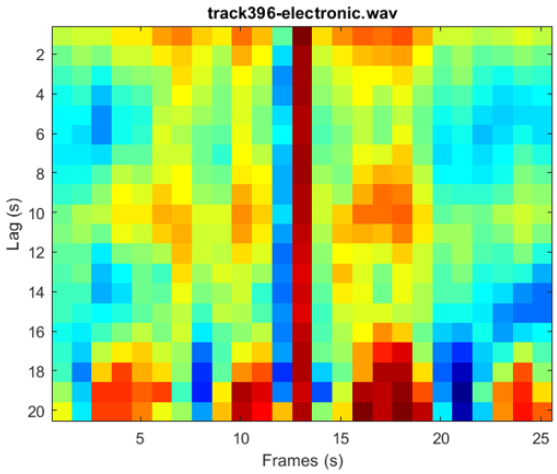
Assignment 10

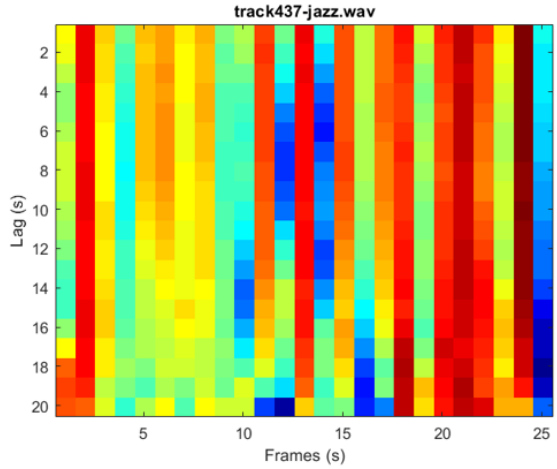
A further evaluation of the rhythmic characteristic of a track is to calculate the autocorrelation of the autocorrelation which gives a sense of how the rhythmic content changes over time. To do this, a small section of the song will be analyzed into a matrix to display. The MATLAB code for this is found in the Assignment 10 Appendix and the plots for the 12 tracks are shown below:

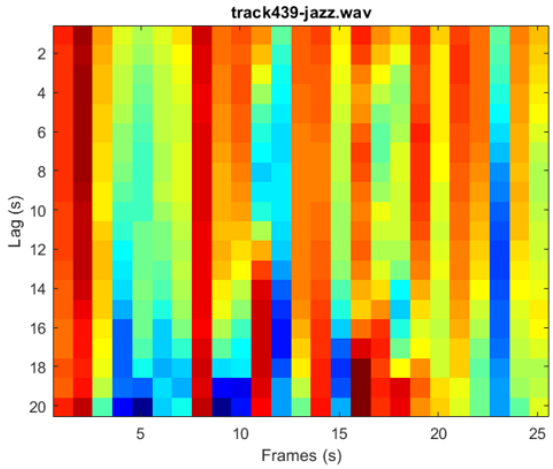


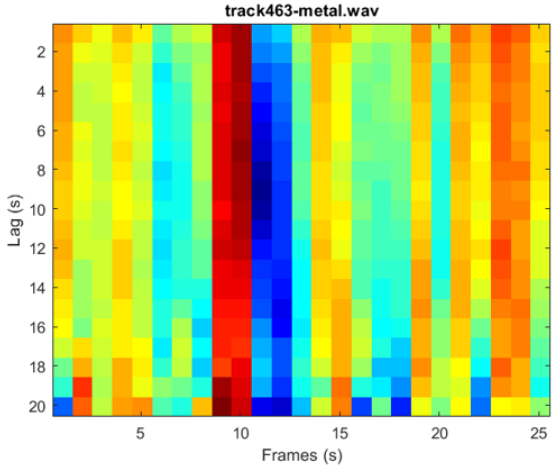


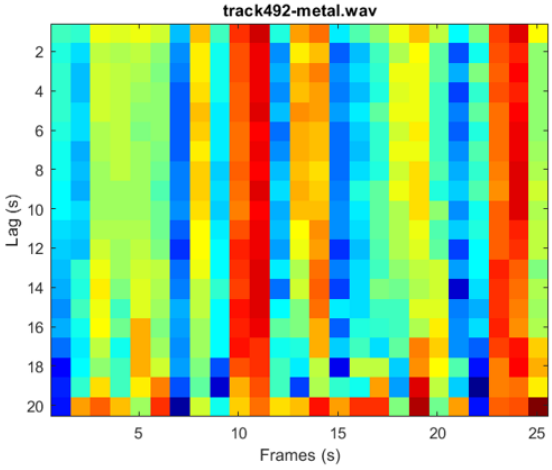


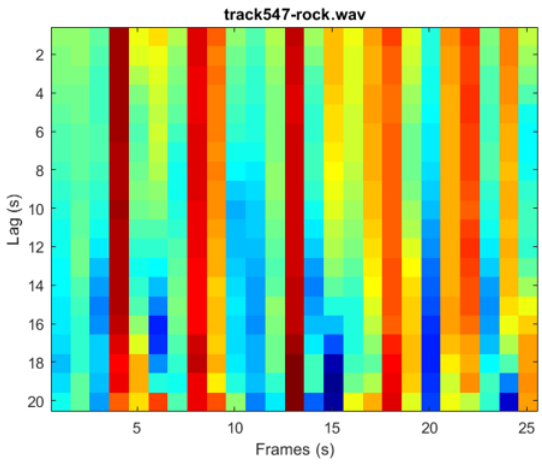


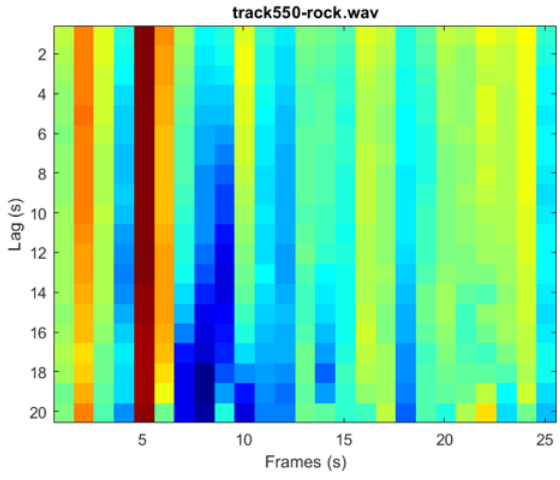

track437-jazz.wav
Lag (s)
Frames (s)

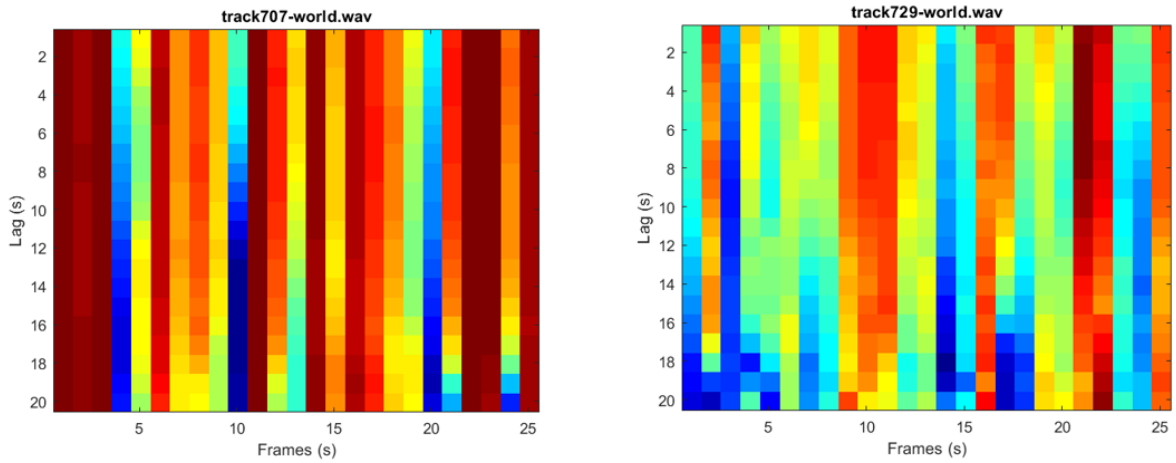

track439-jazz.wav
Lag (s)
Frames (s)


track463-metal.wav
Lag (s)
Frames (s)


track492-metal.wav
Lag (s)
Frames (s)


track547-rock.wav
Lag (s)
Frames (s)

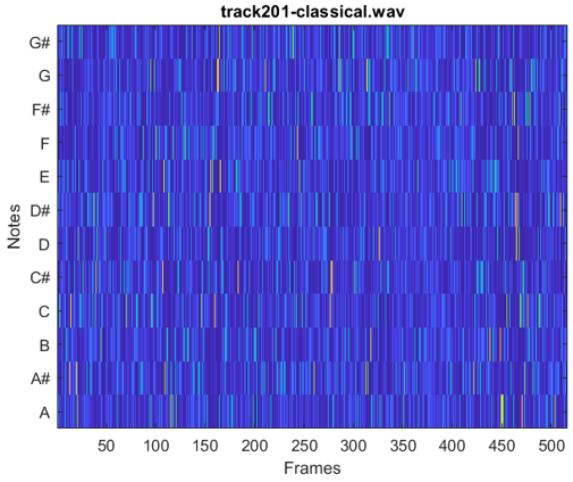

track550-rock.wav
Lag (s)
Frames (s)

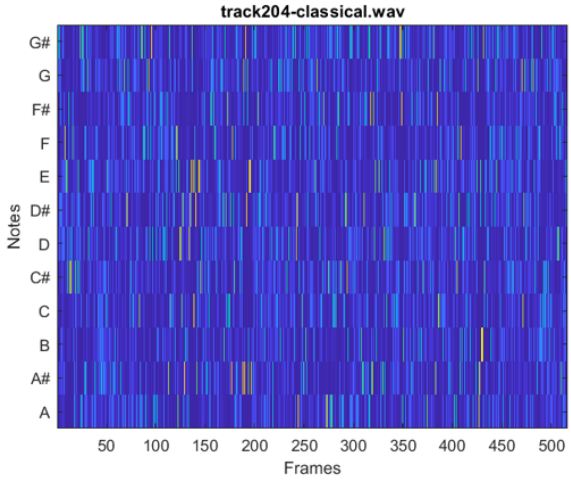


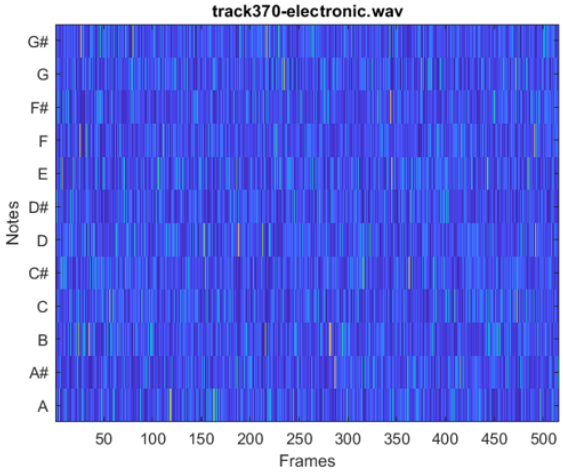
One interesting thing that can be seen from this analysis is the detecting the presence of sustained notes in a track. The World 707 has large strips of high correlation from top to bottom across all lags, which show the note is sustained and similar across many frames and lags. This is present in the track when physically hearing it as well. This sustain can be seen and heard in the jazz tracks as well. The rock and electronic tracks show good examples of a rhythmic pattern because the brighter stripes at evenly frames show strong correlation, indicating a rhythm, which can also be determined from listening.

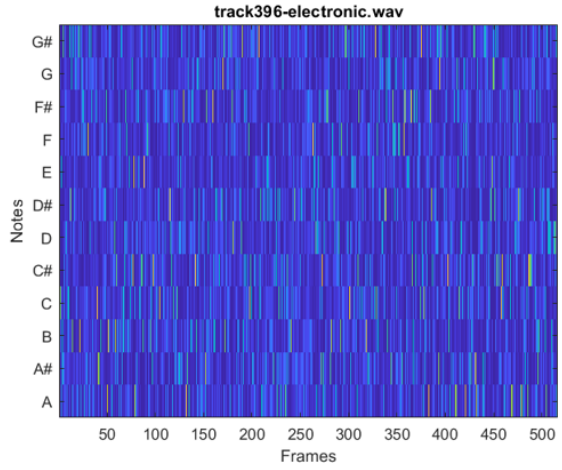
Assignment 11

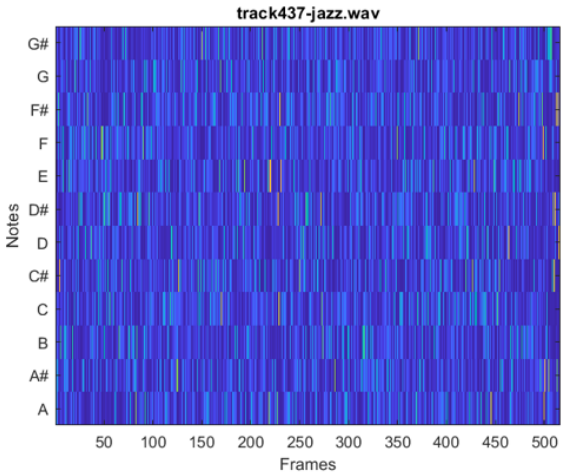
The final analysis to be performed on this lab is to analyze the melodic and harmonic content of a track using chroma detection. This analysis will use the Fourier transform and the detection of local maxima in the frequency domain to map these frequencies to actual notes in the logarithmically spaced tempered scale. The many notes associated with these local maxima throughout a frame of audio give the harmonic content of that particular frame, which can then be used to build a Pitch Class profile for that particular frame, which shows which notes on a tempered scale from A to G# are present and how prominent they are. This would allow another algorithm to determine the key signature and mode of a given piece of music. The MATLAB code for the Pitch Class Profile computation is given in the Assignment 11 Appendix and the chroma plots for the 12 tracks are shown below:

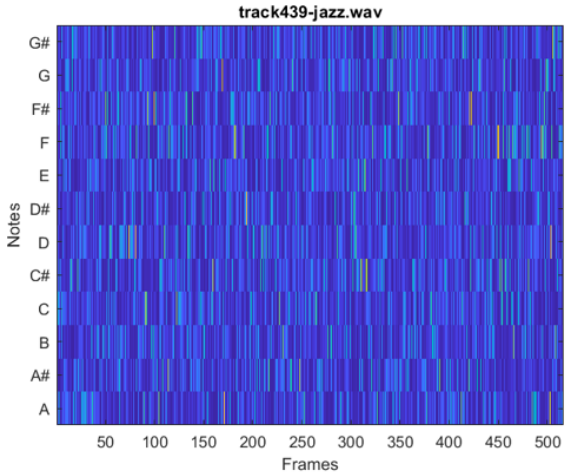

track201-classical.wav
G#
G
F#
F
E
D#
D
C#
C
B
A#
A
Notes
50 100 150 200 250 300 350 400 450 500
Frames

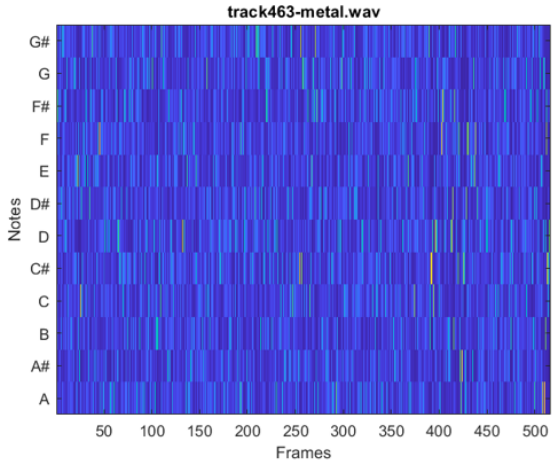

track204-classical.wav
G#
G
F#
F
E
D#
D
C#
C
B
A#
A
Notes
50 100 150 200 250 300 350 400 450 500
Frames

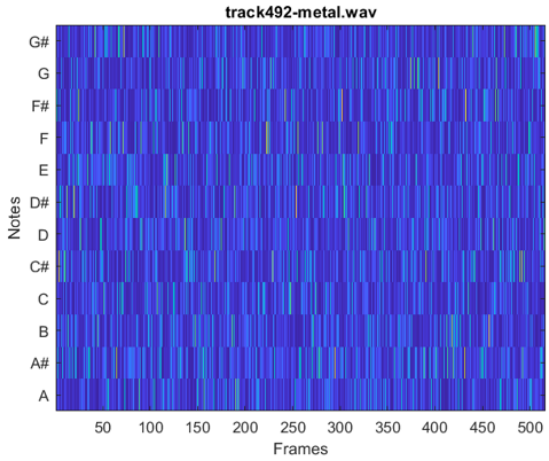

track370-electronic.wav
G#
G
F#
F
E
D#
D
C#
C
B
A#
A
Notes
50 100 150 200 250 300 350 400 450 500
Frames

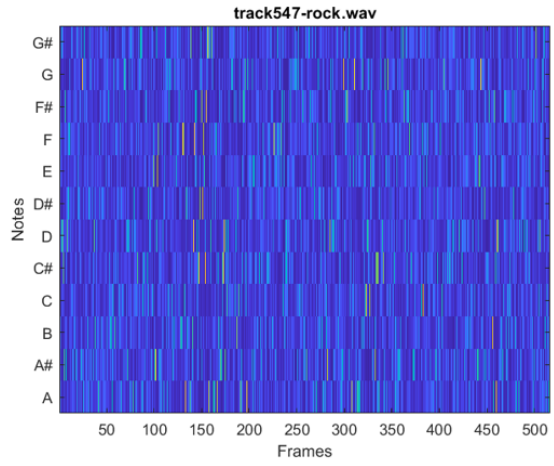

track396-electronic.wav
G#
G
F#
F
E
D#
D
C#
C
B
A#
A
Notes
50 100 150 200 250 300 350 400 450 500
Frames

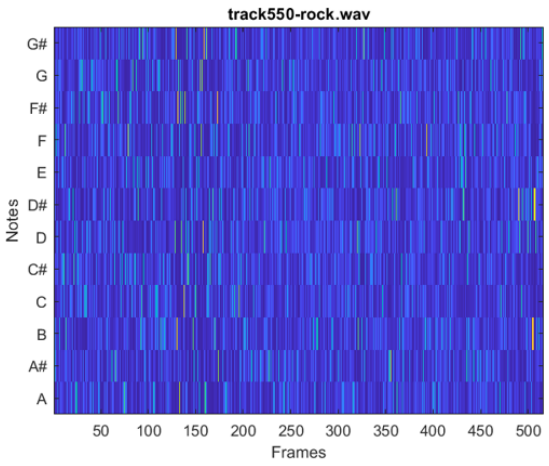

track437-jazz.wav
G#
G
F#
F
E
D#
D
C#
C
B
A#
A
Notes
50 100 150 200 250 300 350 400 450 500
Frames

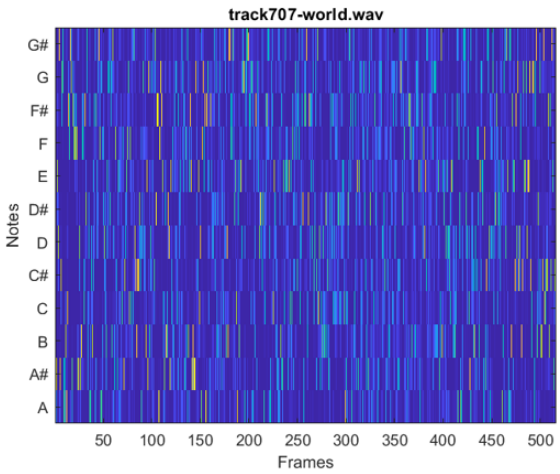

track439-jazz.wav
G#
G
F#
F
E
D#
D
C#
C
B
A#
A
Notes
50 100 150 200 250 300 350 400 450 500
Frames

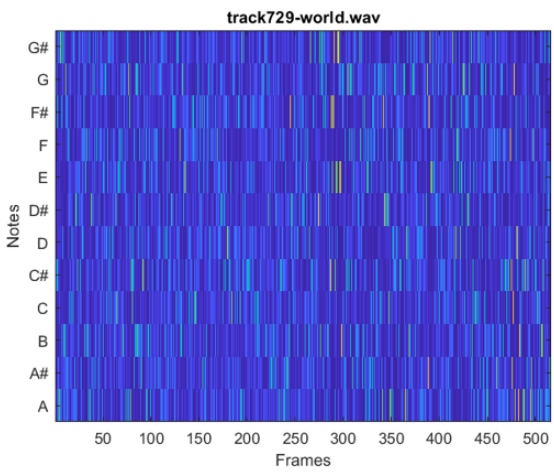

track463-metal.wav
Notes
G#
G
F#
F
E
D#
D
C#
C
B
A#
A
50 100 150 200 250 300 350 400 450 500
Frames


track492-metal.wav
Notes
G#
G
F#
F
E
D#
D
C#
C
B
A#
A
50 100 150 200 250 300 350 400 450 500
Frames


track547-rock.wav
Notes
G#
G
F#
F
E
D#
D
C#
C
B
A#
A
50 100 150 200 250 300 350 400 450 500
Frames


track550-rock.wav
Notes
G#
G
F#
F
E
D#
D
C#
C
B
A#
A
50 100 150 200 250 300 350 400 450 500
Frames


track707-world.wav
Notes
G#
G
F#
F
E
D#
D
C#
C
B
A#
A
50 100 150 200 250 300 350 400 450 500
Frames


track729-world.wav
Notes
G#
G
F#
F
E
D#
D
C#
C
B
A#
A
50 100 150 200 250 300 350 400 450 500
Frames

The chroma plots are difficult to analyze due to the fact that all of the harmonics may not be completely discarded when performing the quantization with peak detection, but many of the strongly present notes in the track can still be seen. For example the classical track 201 is a piano piece composed by Robert Schumann and this portion of the collection is in G major. Upon inspection it can be seen that there is a strong presence of notes in the G major scale, such as the triad of the standard G chord G, B and D, as well as many other notes that could be used within complex chords throughout the piece. Also, the chroma can give information regarding sustain where a signal note holds the same energy over many frames, such as in the world 707 track on the C note around frame 300.

Code Appendix

Lab 2 Driver Script

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clint Olsen
% ECEN 4532: DSP Lab
% Lab 2: Driver Script
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Assignment 1

%Audio File Information
filename = 'track729-world.wav';
N = 512;
info = audioinfo(filename);

%Filter bank info
fbank = cochlearFilterbank(N,filename);
Nb = 40;
K = N/2 + 1;

%Extract 24 seconds from the audio file
track = extractTSeconds(24,filename);

%Variables for MFCC values
mfccdB = zeros(floor(length(track)/N),Nb);
mfcc = zeros(floor(length(track)/N),Nb);

% Find mfcc coef. for each frame in track
for n=1:floor(length(track)/N)
    %Extract N samples based on frame number
    xn = track(1+((N)*(n-1)):1+((N)*(n-1)) + (N-1));

    %Perform the fft and extract the desired components up to K and
    %determine the MFCC coefficients
    Y = abs(fft(xn));
    Xn = Y(1:K);
    [mfccdB(n,:) mfcc(n,:)] = mfccComp(N,Xn,fbank,Nb);
end

dbMax = max(max(mfccdB));
dbMin = min(min(mfccdB));
%% Assignment 2
%Calculate the Spectrum Histogram from the mfcc coef.
hist = spectrumHist(60,-20,81,40,mfccdB);

%% Assignment 3
% Plot the generated histogram
colormap('Jet');
figure;
imagesc((hist));
set(gca, 'YTickLabel', [-20:10:60]) % 20 ticks
xlabel('MFCC Index');
ylabel('dB');
title(filename);

%% Assignment 4
%Generate the similarity matrix
S = similarityMatrix(mfcc);

%Plot the similarity matrix
colormap('Jet');
figure;
imagesc(S);
xlabel('Frames');
ylabel('Frames');
title(filename);

%% Assignment 5

%Generate Rhythm Pattern
Nf = floor(length(track)/N);
B = rhythm1(Nf,S);

%Plot the rhythmic similarity
l = 0:1:Nf-1;
figure;
plot(l,B);
xlabel('Frames');
title(filename);

%% Assignment 6
%Calculate the median of the Similarity Matrix
Sbar = matrixMedian(S);

%% Assignment 7
%Calculate the l values for BPM values between 40 and 240 in steps of 5
BPM = bpm(filename,S,Sbar,N,Nf);
```

```
[maxBPM I] = max(BPM);
trackBPM = 40+(I*5);

%% Assignment 8
%Plot the Normalized BPM vector
bpmHist(BPM, filename);

%% Assignment 9
%Calculate and Plot the AutoCorrelation
rhythm2(Nf,S,filename);

%% Assignment 10
%Calculate the Rhythmic Variation AC
lsize = 20;
msize = 20;
AC = rhythmicVariation(S,lsize,msize,Nf);
ltime = info.SampleRate/(lsize*N);
mtime = info.SampleRate/(msize*N);
colormap('jet')
figure;
imagesc(mtime,ltime,AC);
title(filename);
xlabel('Frames (s)')
ylabel('Lag (s)')

%% Assignment 11
%Calculate the Normamlized pitch class for the track segment
PCP = chroma(track,N);

%% Assignment 12
%Plot the Chroma for the tempered scale
notes = {'G#', 'G', 'F#', 'F', 'E', 'D#', 'D', 'C#', 'C', 'B', 'A#', 'A'};
colormap('jet')
figure;
imagesc(PCP);
set(gca, 'YTick', [1:1:12], 'YTickLabel', notes) % 12 ticks for each note
title(filename);
xlabel('Frames');
ylabel('Notes');
```

Assignment 1

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clint Olsen
% ECEN 4532: DSP Lab
% Lab 2: Assignment 1: MFCC Coefficients to DB
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [mfccdB mfcc] = mfccComp(N, Xn, Hp, Nb)
    %Define K as half of the frame size N
    K = N/2 + 1;

    %Define the vector to hold the coefficients for each filter
    mfcc = zeros(1,Nb);
    mfccdB = zeros(1,Nb);

    %Compute the coefficents for each filter in the bank.
    for p=1:Nb
        for k=1:K
            mfcc(p) = mfcc(p)+(abs(Hp(p,k)*Xn(k)))^2;
        end
        mfccdB(p) = 50 + 10*log10(mfcc(p)); %Convert to dB
    end

end
```

Assignment 2

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clint Olsen
% ECEN 4532: DSP Lab
% Lab 2: Assignment 2: Spectrum Histogram
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function hist = spectrumHist(maxdB, mindB, rows,cols, mfccdB)
    %Declare the power levels x filter banks count matrix for histogram (81 possible dB values, 40
    %filter banks)
    hist = zeros(rows,cols);

    %Loop through mfcc coefficients to find power levels
    for i=1:length(mfccdB)
       for j=1:length(mfccdB(1,:))
           val = round(mfccdB(i,j));
           if val < mindB %Assign to min dB level
               hist((mindB+(-mindB+1)),j) = hist((mindB+(-mindB+1)),j) + 1;
           elseif val > maxdB %Assign to max dB level
               hist((maxdB+(-mindB)+1),j) = hist((maxdB+(-mindB)+1),j) + 1;
```

```
            else
                hist((val+(-mindB+1)),j) = hist((val+(-mindB+1)),j) + 1;
            end
        end
    end

    %Normalize the sum of each column to 1
    for i=1:length(mfccdB(1,:))
        val = sum(hist(:,i));
        hist(:,i) = hist(:,i) ./ val;
    end
end
```

Assignment 4

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clint Olsen
% ECEN 4532: DSP Lab
% Lab 2: Assignment 4: Similarity Matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function simMatrix = similarityMatrix(mfcc)
    %Declare the similarity matrix
    simMatrix = zeros(length(mfcc),length(mfcc));

    %Calculate the correlation between each frame in track through the dot
    %product
    for i=1:length(mfcc)
        for j=1:length(mfcc)
            simMatrix(i,j) = ((dot(mfcc(i,:),mfcc(j,:)))/(norm(mfcc(i,:))*norm(mfcc(j,:))));
        end
    end
end
```

Assignment 5

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clint Olsen
% ECEN 4532: DSP Lab
% Lab 2: Assignment 5: Rhythmic Pattern 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function B = rhythm1(Nf, S)
    %Declare the Rhythm Martix
    B = zeros(1,Nf);

    for l=0:(Nf-1)
        for n=1:Nf-l
            B(l+1) = B(l+1) + S(n,n+l); %Count diagonal at different lags
        end
        B(l+1) = (1/(Nf-l))*B(l+1); %Complete computation
    end
end
```

Assignment 6

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clint Olsen
% ECEN 4532: DSP Lab
% Lab 2: Assignment 6: Median of Matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Sbar = matrixMedian(S)
    %Calculate the Median of Matrix S
    Sbar = median(S(:));
end
```

Assignment 7

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clint Olsen
% ECEN 4532: DSP Lab
% Lab 2: Assignment 7: BPM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function BPM = bpm(filename,S, Sbar, N, Nf)
    %Declare the lag array and bpm array
    lag = zeros(1,41);
    bpm = 40:5:240; % Space BPM from 40-240 at steps of 5 BPM
    BPM = zeros(1,41);

    %Extract sampling freq
```

```
    info = audioinfo(filename);
    Fs = info.SampleRate;

    %Calculate the lag values for each BPM
    lag = (60*Fs) .* (1./(bpm.*N));

    %Calculate the number of times S(i,i+l) is larger than Sbar
    for l=1:length(lag)
        for i=1:(Nf-round(lag(l)))
            if S(i,i+round(lag(l))) > Sbar
                BPM(l) = BPM(l) + 1;
            end
        end
    end
end
```

Assignment 8

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clint Olsen
% ECEN 4532: DSP Lab
% Lab 2: Assignment 8: BPM Histogram
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function PH = bpmHist(bpm,filename)
   %Normalize the BPM vector
    val = sum(bpm);
    PH = bpm ./ val;

    %Plot this normalized BPM vector between 40 and 240 BPM
    figure;
    plot(40:5:240,PH);
    title(filename);
    xlabel('BPM');
end
```

Assignment 9

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clint Olsen
% ECEN 4532: DSP Lab
% Lab 2: Assignment 9: Rhythmic Pattern 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function ac = rhythm2(Nf, S,filename)
    %Declare the Rhythm Martix Autocorrelation
    ac = zeros(length(S(:,1)),Nf);
    AC = zeros(1,Nf);

    %Calculate the Autocorrelation
    for i=1:length(S(:,1))
        for l=0:Nf-1
            for j=1:Nf-l
                ac(i,l+1) = ac(i,l+1) + S(i,j)*S(i,j+l);
            end
            ac(i,l+1) = (1/(Nf-l))*ac(i,l+1);
        end
    end

    %Normalize the Autocorrelation
    for l=1:Nf
        for i=1:Nf
            AC(l) = AC(l) + ac(i,l);
        end
    end

    %Plot the AC
    figure;
    plot(0:1:Nf-1,AC);
    xlabel('Frames');
    title(filename);

end
```

Assignment 10

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clint Olsen
% ECEN 4532: DSP Lab
% Lab 2: Assignment 10: Rhythmic Variation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function AC = rhythmicVariation(S, lsize, msize, Nf)
    %Declare variable to hold AC matrix
```

```
    AC = zeros(lsize,floor(Nf/msize));

    %Perform the double summation for the AC on itself
    for l=0:lsize-1
        for m=0:(floor(Nf/msize)-1)
            AC(l+1,m+1) = 1/(msize*(msize-l))*sum(sum(S((1:msize) + msize*m, (1:msize-l) + msize*m) ...
                .*S((1:msize) + msize*m, (1:msize-l) + msize*m + l)));
        end
    end
end
```

Assignment 11

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clint Olsen
% ECEN 4532: DSP Lab
% Lab 2: Assignment 11: Pitch Class Profile
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function PCP = chroma(track, N)
    % Frequency of the lowest octave A on piano
    f0 = 27.5;
    K = N/2 + 1;
    PCP = zeros(12,floor(length(track)/N));

    for n=1:floor(length(track)/N)
        %Extract N samples based on frame number
        xn = track(1+((N)*(n-1)):1+((N)*(n-1)) + (N-1));

        %Perform the fft and extract the desired components up to K
        Y = abs(fft(kaiser(N,0.5).*xn));
        Xn = Y(1:K);

        %Find the local maxima values and locations
        [fk I] = findpeaks(Xn);

        %Find the nearest semitones and map to corresponding note,
        %regarless of octave
        PL = floor(12*log2(fk./f0));
        PH = ceil(12*log2(fk./f0));
        CL = mod(PL,12);
        CH = mod(PH,12);

        %Create the weighted sum of the semitones for each frame
        rL = 12*log2(fk./f0) - PL;
        wL = (cos(pi.*rL./2)).^2;
        rH = 12*log2(fk./f0) - PH;
        wH = (cos(pi.*rH./2)).^2;

        %Calculate the PCP values
        for k=1:length(CH)
            PCP(CL(k)+1,n) = PCP(CL(k)+1,n) + wL(k)*(Xn(k))^2;
            PCP(CH(k)+1,n) = PCP(CH(k)+1,n) + wH(k)*(Xn(k))^2;
        end

        %Normalize the column vector over all semitones
        PCP(:,n) = PCP(:,n)./(sum(PCP((1:12),n)));
    end
end
```