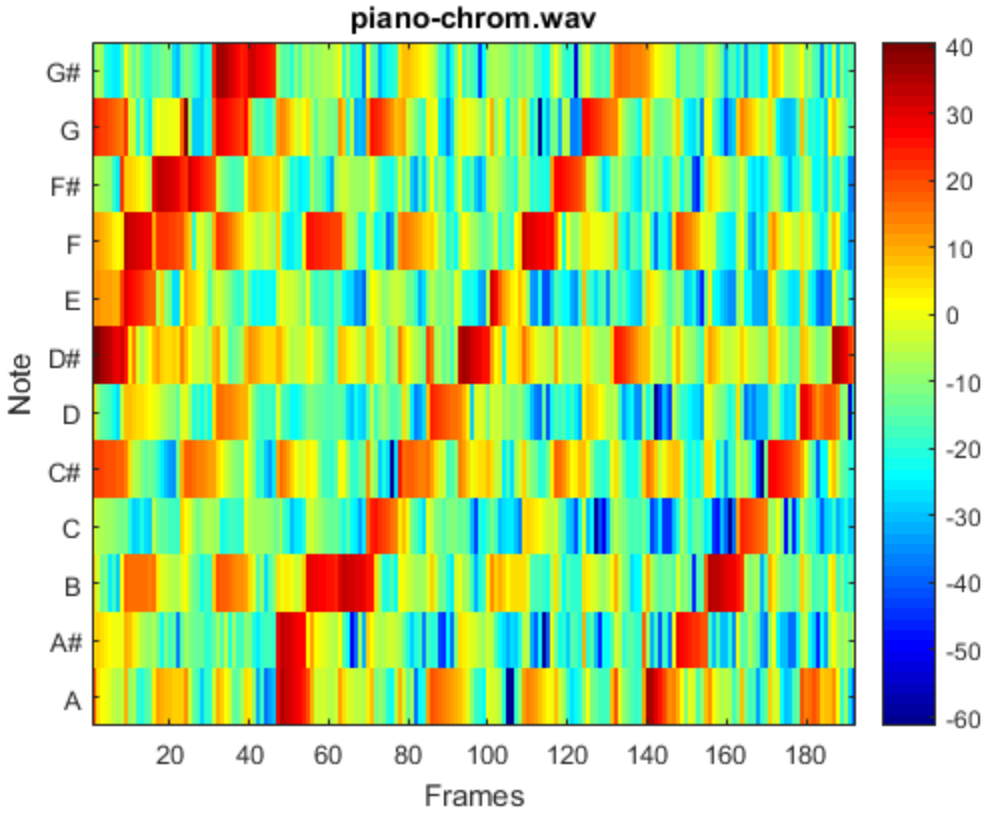
Digital Signal Processing Lab 2



By Zhi Jie Huang
2/20/17

1. Introduction:

This is the second lab on Digital Signal Processing, the purpose of this lab is to explore the methods for content-based music information retrieval based on the MFCC coefficient calculated from the Lab1. In this lab, we will get a chance to explore the musical features such as rhythm and tonality.

2. Calculation of similarity matrix:

In data analysis, similarity matrix is a graphical representation of similar sequence in data series. The similarity matrix measures the cosine of the angle between the spectral signatures of frames i and j from the mfcc matrix. If frame i and j is very similar, the similarity matrix value should be very close to 1(i.e very close to red); if frame I and j is not similar, the similarity matrix value should be very close to 0(i.e very close to blue).

As you can see, the two classical is mostly blue and green, that means the classical music doesn't have a lot of similar note been play during different note. As we know, classical music has wide spread of notes been played throughout the song. So it physically make sense.

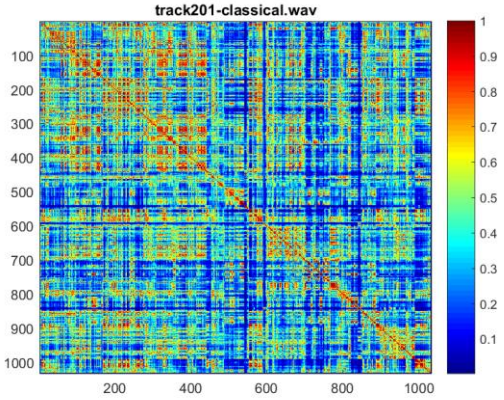


Fig 1: Similarity matrix for track201

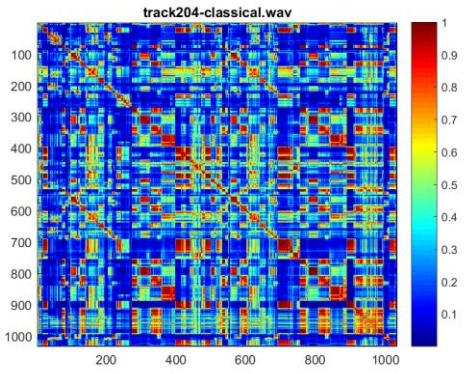


Fig 2: Similarity matrix for track 204

Comparing the two electronics music, if you listen to track-370 electronic, you can feels a lot of different notes has been played through out the song, there aren't a lot of repetitive note been played, therefore the similarity matrix is mostly blue. However, if you listen to track 396-electronics, you can tell there are multiple notes being play repetitive throughout different frames. Therefore, there are a lot of values closer to 1(red).

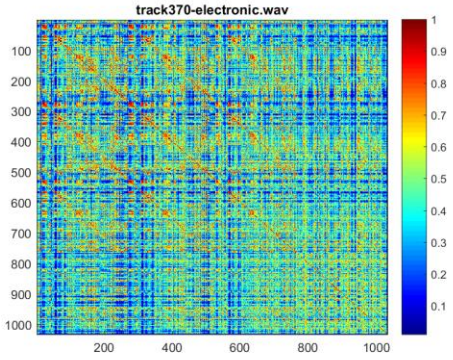


Fig 3: Similarity matrix for track370

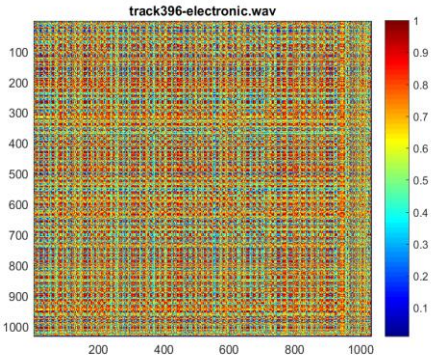


Fig 4: Similarity matrix for track 396

Jazz music encompasses a wide range of music spanning a period of over 100 years, so it does have some similar note played throughout song for both of the track. So the frames are very similar between the i frame and the j frame.

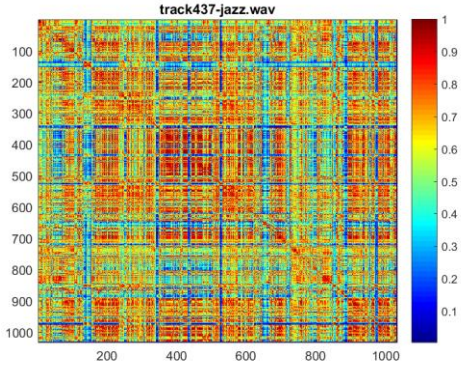


Fig 5: Similarity matrix track 437

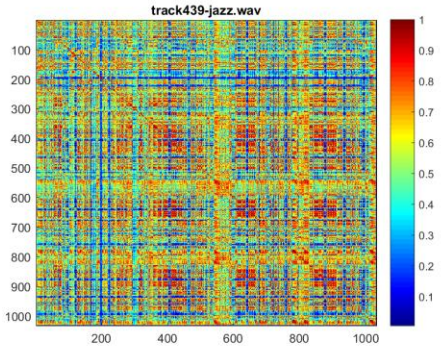


Fig 6: Similarity matrix track 439

If you listen to track 463, there is a certain musical rhythm going on throughout the song, therefore, you can see notes been play every other 100 frames. However, for track 492 metal, there are too many notes been played repetitively, so it's hard to distinguish a pattern for the song. The similarity matrix value for track 492 are all very close to each other.

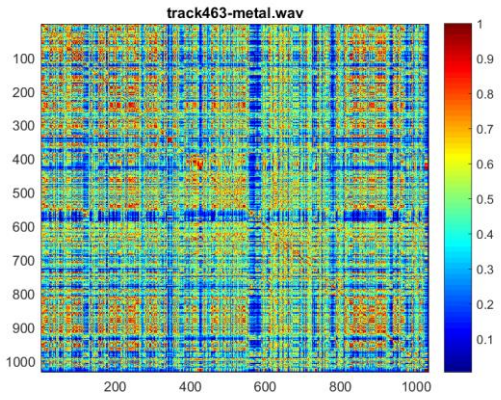


Fig 7: Similarity matrix track 463

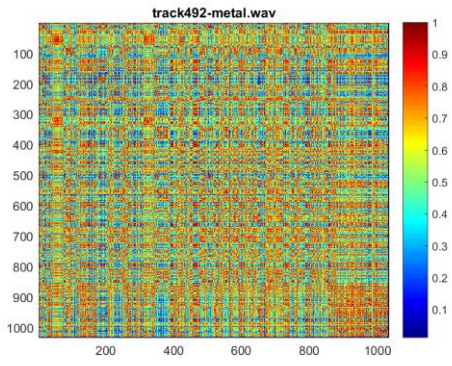


Fig 8: Similarity matrix track 492

At the beginning of the track 547 rock, there are a lot of similar note been play. You can hear it from the middle of the song and see it in figure 9. For track 550-rock, while I was listening to the track, the notes been played doesn't seems to have a repetitive pattern, therefore you can see from figure 10, the value for the similarity matrix is around 0.4~0.6.

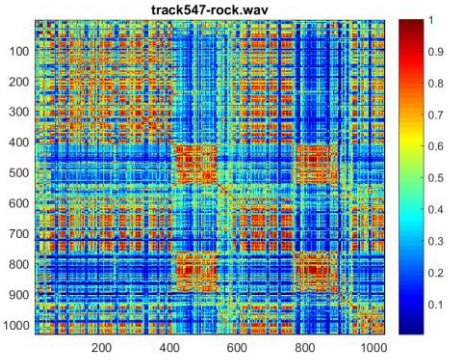


Fig 9: Similarity matrix track 547

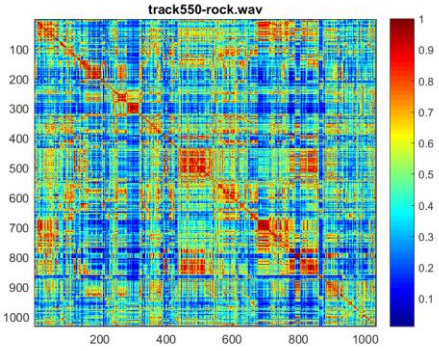


Fig 10: Similarity matrix track 550

When you are listening to track 707, it is really quiet, and it doesn't have a lot of similar note been play， except at frame 200, there was a flute playing for a little, it doesn't have a lot of variation of note, therefore, you can see a lot of red in the middle. When you are listening to the track 729, it have very monotonous, it doesn't have a lot of similar note play thoughout the song.

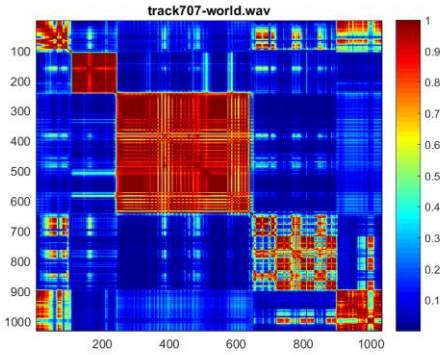


Fig 11: Similarity matrix track 707

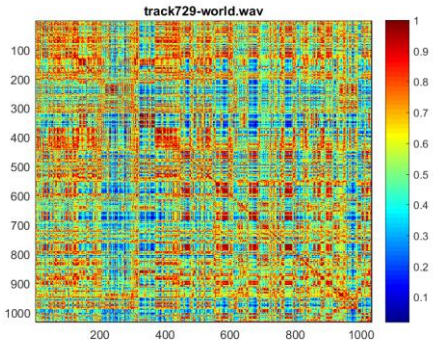


Fig 12: Similarity matrix track 729

3. A first estimate of the rhythm

In this part of the lab, we are trying to use the similarity matrix information we calculated from part 2 to detect the presence of patterns that repeated every l frames away. The B(l) is the sum of all the entries on the lth diagonal.

If you listen to track 201, you can tell there isn't a lot of strong rhythm. It match what I have in the fig 13. For track 204, it has a little more distinct rhythm than the track 201, because it's mainly piano, it couldn't detect rhythm.

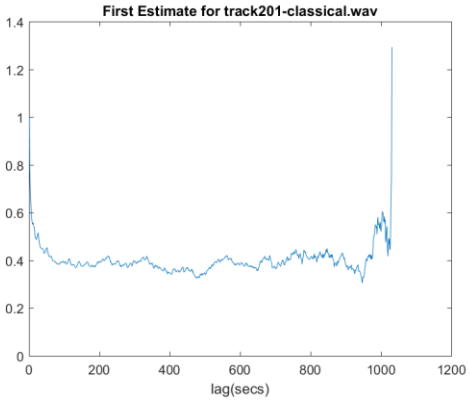


Fig 13:first estimate for track 201

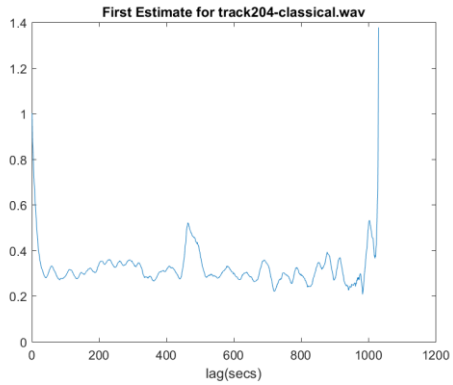


Fig 14:first estimate for track 204

For track 370, it doesn't have a very strong rhythm, it has a few pulses in the in the middle of the track. For track 396, it has a very strong rhythm. If you listen to the track, it has a dubstep playing as a background music, therefore it has a very strong rhythm.

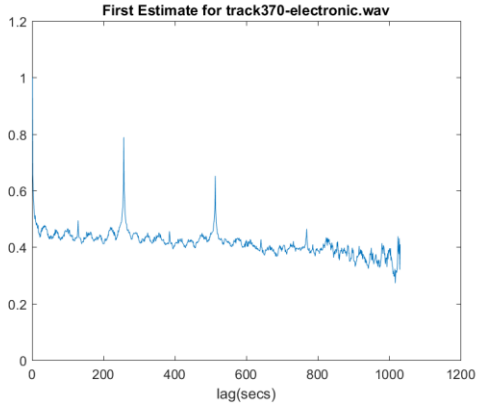


Fig 15: first estimate for track 370

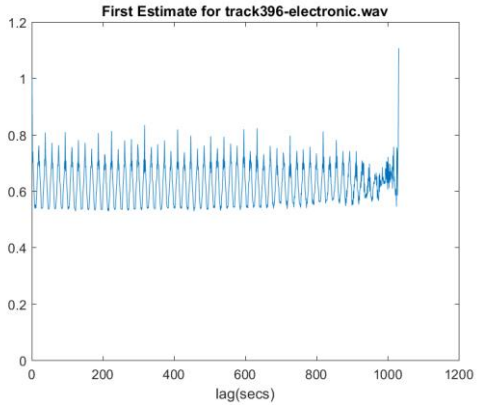


Fig 16:first estimate for track 396

For both track 437 and track 439, it has a very weak rhythmic structure, it doesn't fluctuate too much like you can see from figure 17 and 18.

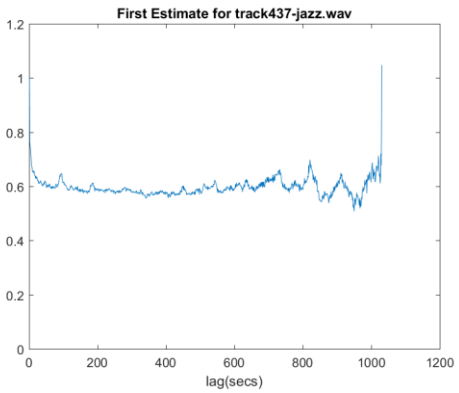


Fig 17: first estimate for track 437

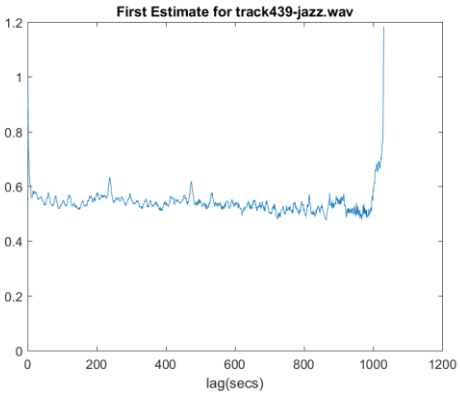


Fig 18:first estimate for track 439

For track 463, you can hear the drum playing in the background. It gives a very strong rhythmic structure to track 463. As you can see below in figure 19, it shows a very steady rhythm going throughout the whole song. The track 493 does sounds like it has a very strong rhythm, however, from the first estimate of the song, it doesn't seems like it has a very strong rhythmic structure.

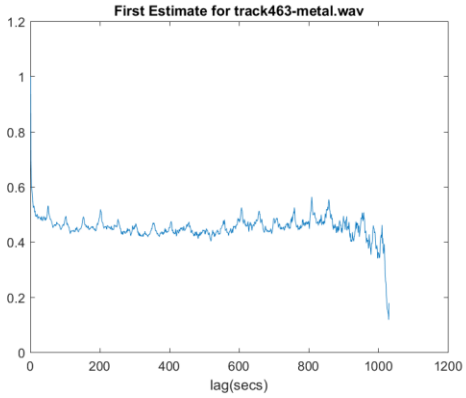


Fig 19: first estimate for track 463

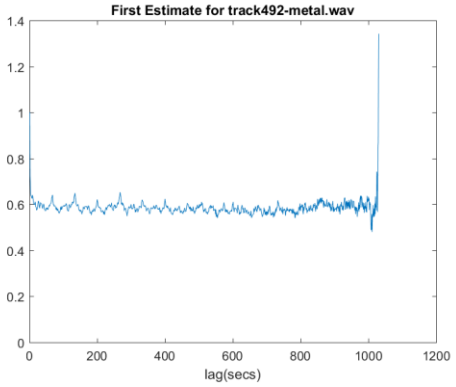


Fig 20:first estimate for track 492

For track 547, it has a electronic guitar playing as a background music, therefore, it has a pretty strong rhythmic structure. For track 550, when I was listing to it, it has a weak rhythmic structure, it matches what I have in figure 22.

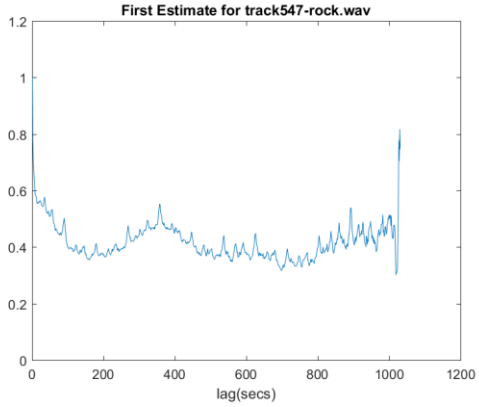


Fig 21: first estimate for track 547

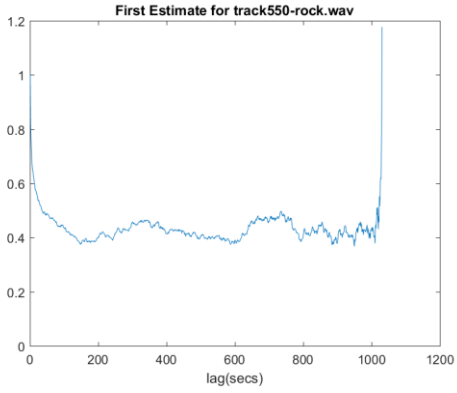


Fig 22: first estimate for track 550

For track 707, it has a very weak rhythmic structure, however, since it is played with one instrument at a time and gradually raising its tone. As you can see in figure 23, it was raising and falling very smoothly. For track 729, it sounds very soft, it doesn't sounds like it has too much rhythm. See figure 24, you can see it.

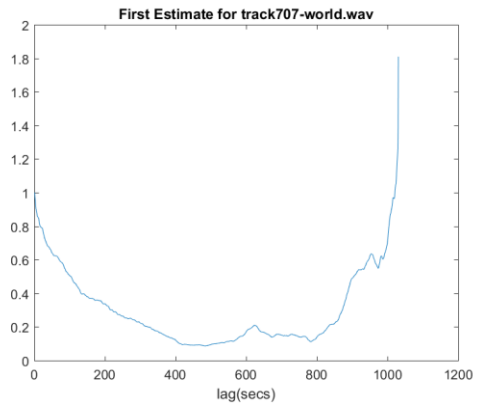


Fig 23: first estimate for track 707

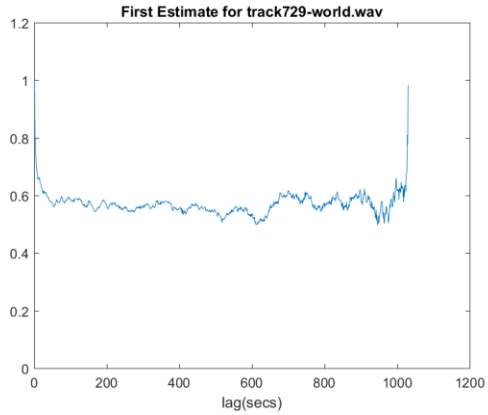


Fig 24: first estimate for track 729

4. A better estimate of the rhythm

Rather than compounding the similarity matrix between frames that are at a fixed lag apart, in this part of the lab, we are going to try to find the similarity of frame I and j will repeat at j+l frames.

Comparing the first estimation of track 201 and track 204, you can tell especially from track 204, there are so much more rhythm , it provides us a better estimation for the rhythm. If you listen to track 201, you can tell there isn't a lot of strong rhythm. It match what I have in the fig 13. For track 204, it has a little more distinct rhythm than the track 201, it can detect piano rhythms.

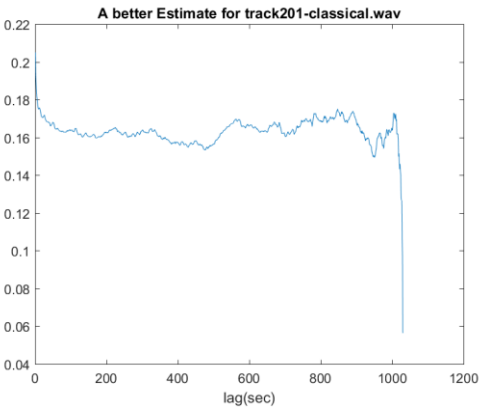


Fig 25: better estimate for track 201

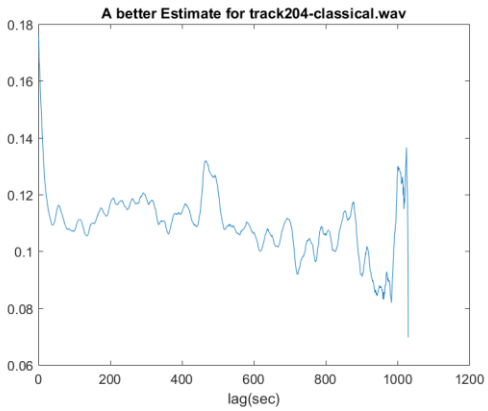


Fig 26: better estimate for track 204

For the track 370 and 396, the better estimate has a cleaner rhythmic structure than the first estimate. For track 370, it doesn't have a very strong rhythm, it has a few pulses in the in the middle of the track. For track 396, it has a very strong rhythm.

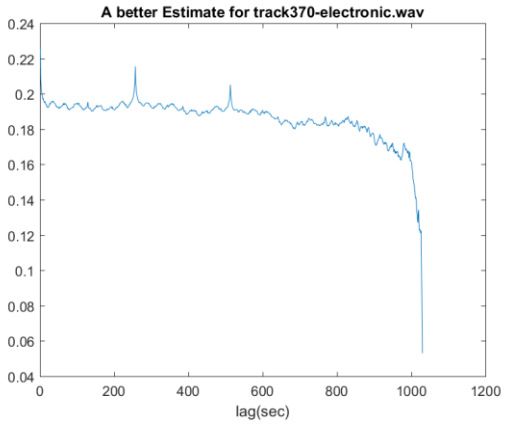


Fig 27: better estimate for track 370

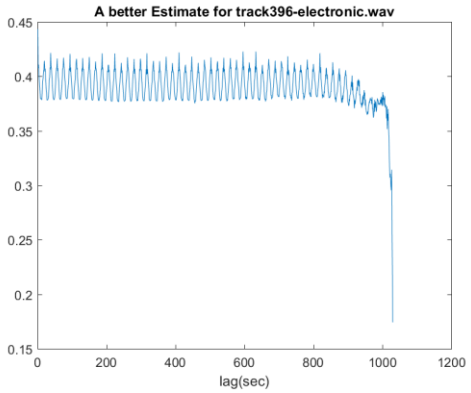


Fig 28: better estimate for track 396

From the better estimate for track 437 and track 439, it could detect the rhythm better than the first estimate, you can see more fluctuation in figure 30 than figure 18. For both track 437 and track 439, it has a very weak rhythmic structure, it doesn't fluctuate too much like you can see from figure 17 and 18.

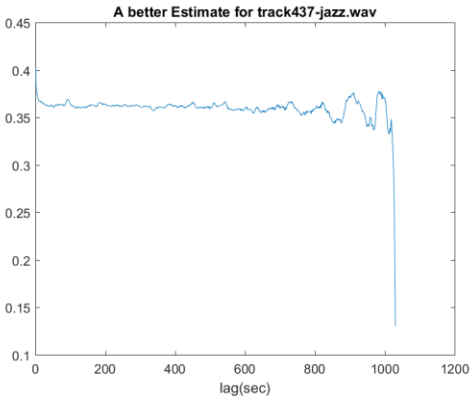


Fig 29: better estimate for track 437

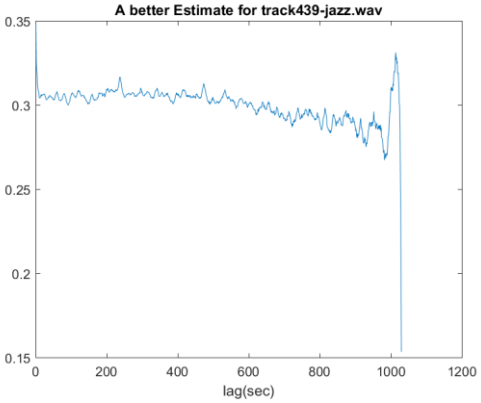


Fig 30: better estimate for track 439

From the better estimate from the track 463 and 492, it pretty much stayed the same as the first estimate. It both has weak rhythm structure.

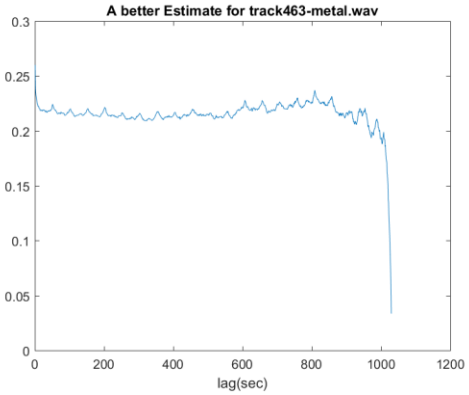


Fig 31: better estimate for track 463

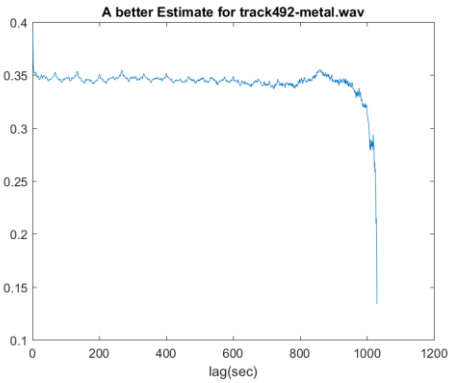


Fig 32: better estimate for track 492

From the better estimate of track 547, it match even better as it has a electronic guitar playing as a background music, therefore, it has a pretty strong rhythmic structure. For track 550, it has a weak rhythmic structure, it matches what I have in figure 34.

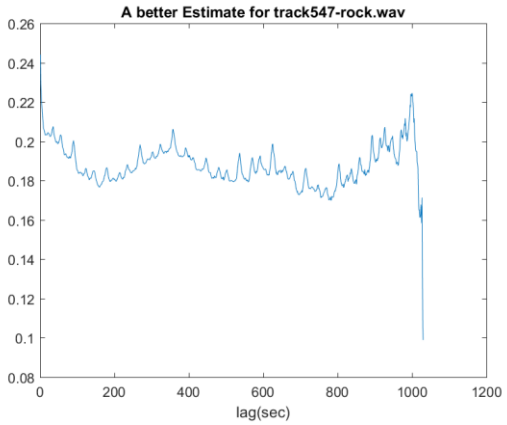


Fig 33: better estimate for track 547

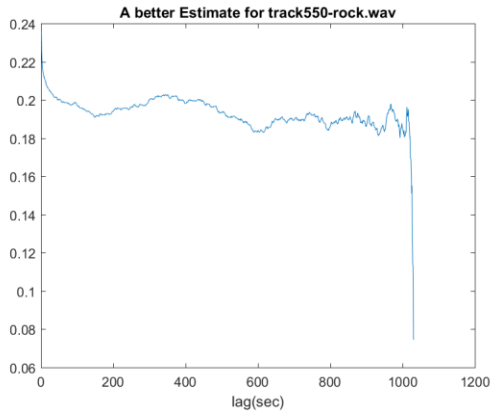


Fig 34: better estimate for track 550

From the better estimate for track 707 and track 729, we can see it's pretty much the same as the first estimate. For track 707, it has a very weak rhythmic structure, because it is played with one instrument at a time and gradually raising its tone. As you can see in figure 23, it was raising and falling very smoothly. For track 729, it sounds very soft, it doesn't sounds like it has too much rhythm.

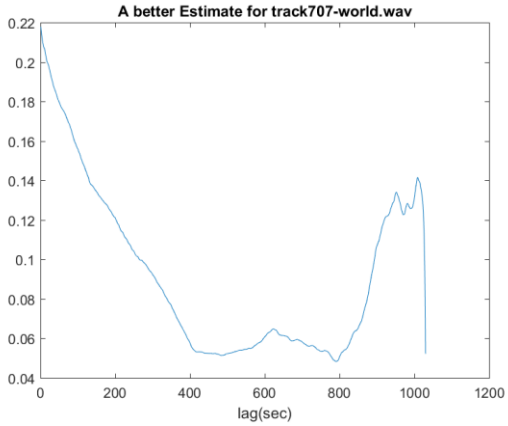


Fig 35: better estimate for track 707

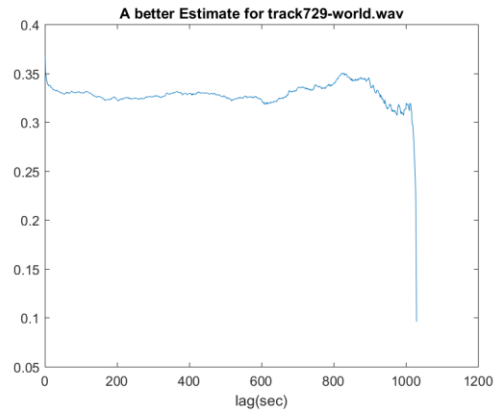


Fig 36: better estimate for track 729

5. Computation of the rhythm index

In this part of the lab, we are interested in calculating the dynamic changes in the rhythm for the songs. Therefore, we are going to use a short time windows formed by 20 frames(or 1 second) over which we calculate the Autocorrelation vector.

For track 201, it doesn't have too much various rhythm pattern, you can see visually from figure 37, it is mostly blue. For track 204, it does have a lot of variation of rhythm patterns, as you can tell from the better estimate for track 204, it shows the fluctuation of the rhythm for track 204, this image is a more visual representation of that.

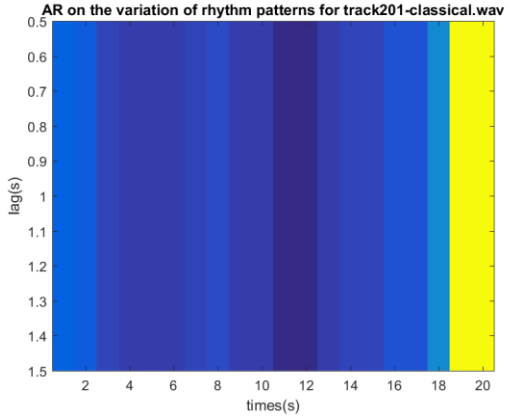


Fig 37: AR for track 201

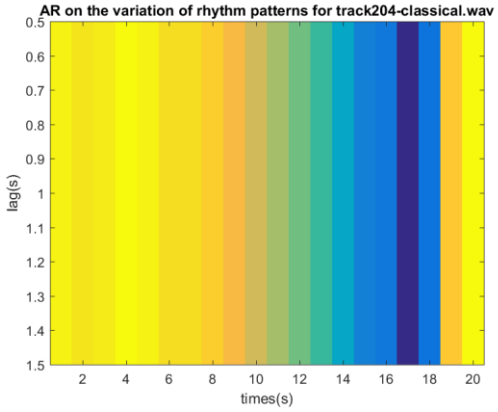


Fig 38: AR for track 204

For track 370, you can see from the color change of figure 39, there are a lot of different rhythm pattern, however it will gradually transition into it. For track 396, it is a electronic song with a strong rhythm patterns, however, it is mostly the same pattern, so it's mostly blue.

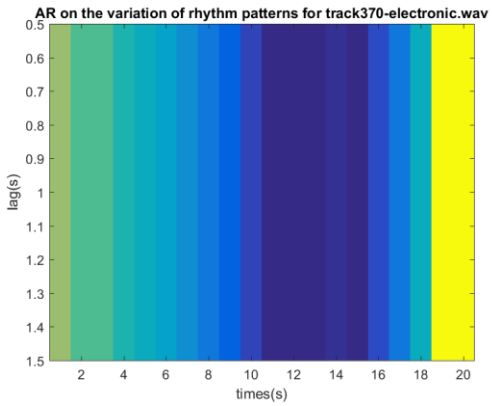


Fig 39: AR for track 370

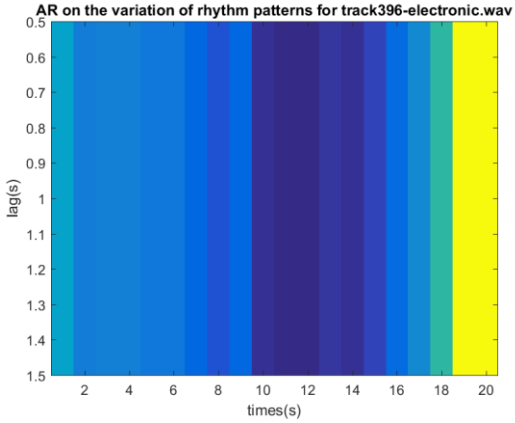


Fig 40: AR for track 396

From both track 437 and 439, you can tell there isn't a lot of variation of rhythm pattern for this track, you can also take a closer look at the better estimate of the rhythm, it almost like a straight like going across.

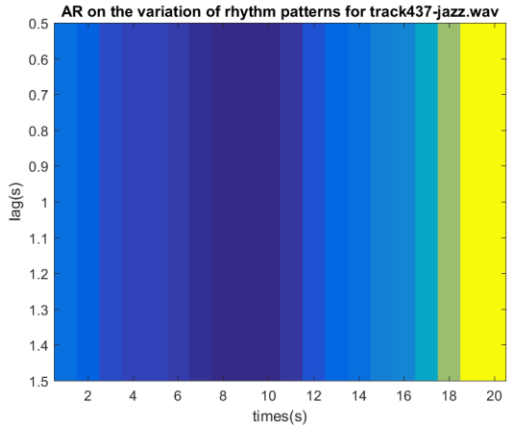


Fig 41: AR for track 437

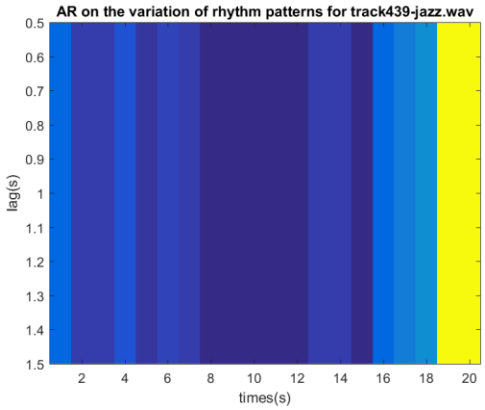


Fig 42: AR for track 439

For track 463 and 492, it has a little more fluctuation in rhythm than track 437 and 439, it shows more color different from figure 43 and 44 to figure 41 and figure 42. However, it doesn't have too much of variation of rhythm patterns.

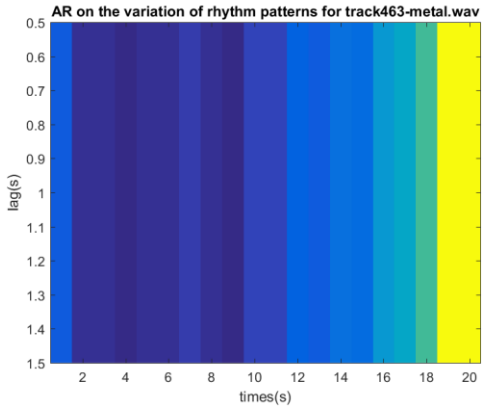


Fig 43: AR for track 463

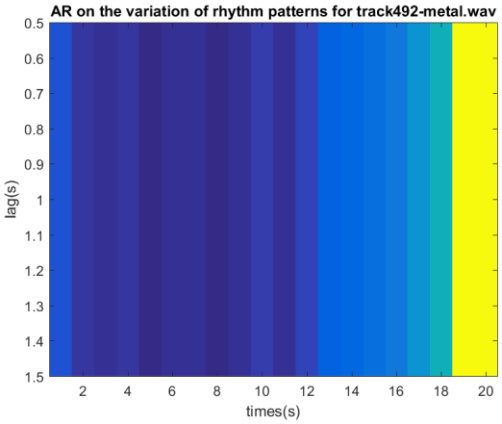


Fig 44: AR for track 492

For track 547, it has a very strong rhythmic pattern, however it's pretty much the same rhythm pattern, it doesn't really vary, so therefore, when we are measuring the variation of the rhythm pattern, it is mostly blue. For track 550, it has a very weak rhythm pattern and it doesn't vary too much, therefore it's mostly blue.

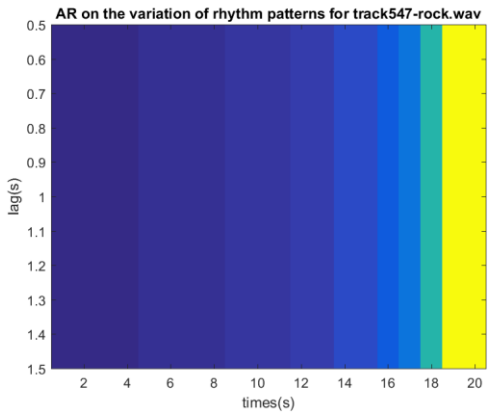


Fig 45: AR for track 547

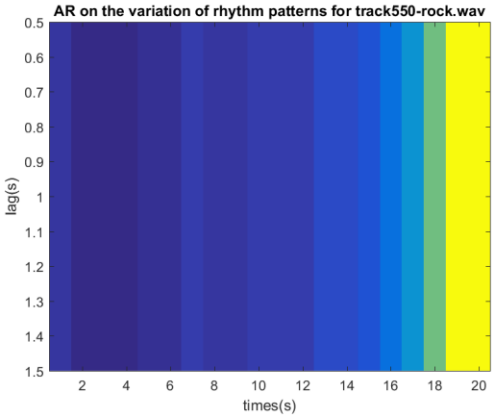


Fig 46: AR for track 550

For track 707, it doesn't really have a rhythm pattern, therefore it's mostly blue.
For track 729, it doesn't have a variation of rhythm pattern, so it's mostly blue.

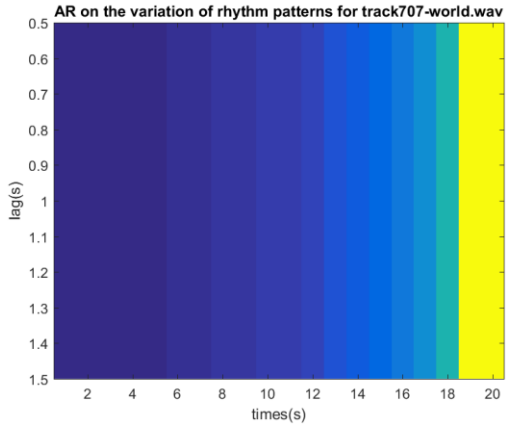


Fig 47: AR for track 707

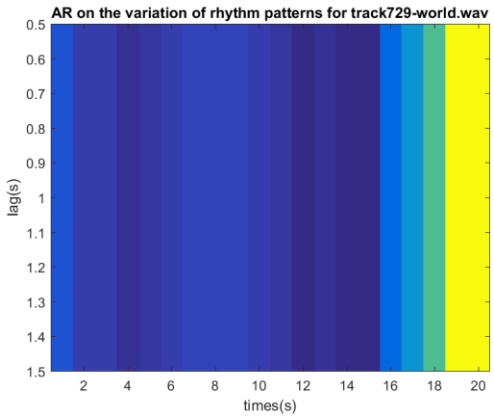


Fig 48: AR for track 729

6. Normalized Pitch Class Profile

In this part of the final project, I am trying to first find corresponding frequency of the local maxima of the song, bring all of those peak frequencies into the first octave with different semitones, based on the distance the semitones that is away from the octave; I distributed the peak frequencies with the weighted function. And finally I square the peak value of the peak frequencies and sum up the energy of each note.

The way my algorithm works is I find the maximum value of the whole song, use a value that is 23 db below my maximum value as a threshold, any value below the threshold get filter out.The way to calculate the NPCP(Normalized Pitch Class Profile) is to pass in 2048 samples at a time into my NPCP.m, in addition to that, I move the frame of the song at half of the frame which it's 1024 samples at a time. Because I want my signal to overlapped in order to prevent the loss of information. I

am using the hann window to cookie-cut my signal.I did a Fast Fourier transform on the 1024 samples of my signals multiply by the hann window in order to take the signal into frequency domain. Since the fft will give us half positive frequency and half negative frequency, I truncate the negative frequency of the fft signal by taking the first signal up to its nyquist frequency (i.e. 2048/2+1 = 1025 in this case). With the fourier transform of the signal, since some of the value of the fourier transform can be negative. I took the absolute value of the fourier transform, find the local maxima of the song that is within the range of -23db of the maximum peak of the song. After I have the peak frequency, since the fk is in representation of the frame index, therefore, I did conversion to frequency. Bring all the frequencies to semitones, and then we divide each semitone by 12 and take the remainder of the division. That gives me the chroma of the weighted sum of all the peak frequencies into the first octave. Since matlab have index starting at 1 instead of 0, therefore, I added 1 to all the note. Since while we calculate the semitone, I did some rounding, therefore, since 12*log2(FK/f0) is always between 1 and -1 of the semitones. After I mapped all the peak frequencies into different note, since I don't have time to look at each note one by one, therefore, I sum up the total energy of each note on all of the octave and multiply the peak square by the weighted function.

The scale on the right of the graph represents the decibel of the particular notes being played. When color is represented the amplitude on the graph, it means that the note on a scale of A-G is being played. Most easily followed along with songs that have a single instrument playing a melody, the notes of the music being played can be visually read on the graph. For instance, when I was looking at the piano-chrom.wav, there is only a single note been played at a time, I can see clearly the pianist started with D# and going up in frequency. Since all octave has been bring into the first octave, when the music hit the next octave,it will just start with A again.

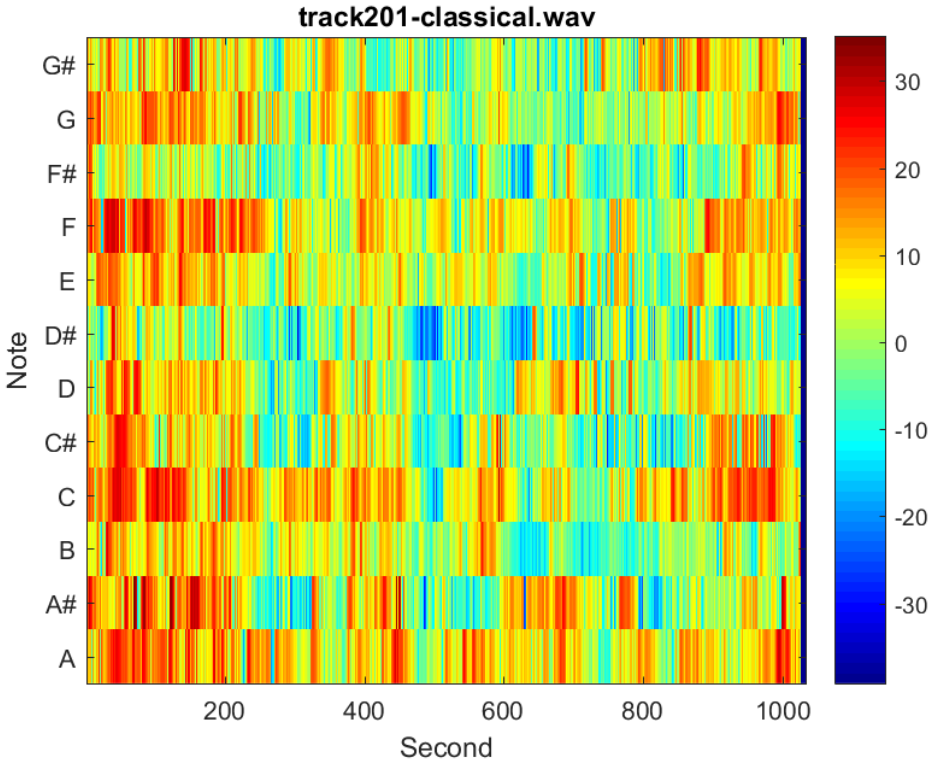


Fig 49: NPCP for track 201

The song Violin Concerto in A Minor is in the key of A minor, which has a tonal center of the most frequently occurring note A. The graph shows A as the most common note in the song. The concerto has multiple instruments playing different parts and notes, therefore the graph corresponds with its representation of many notes being played simultaneously. I can hear a lot of harmony on this track.

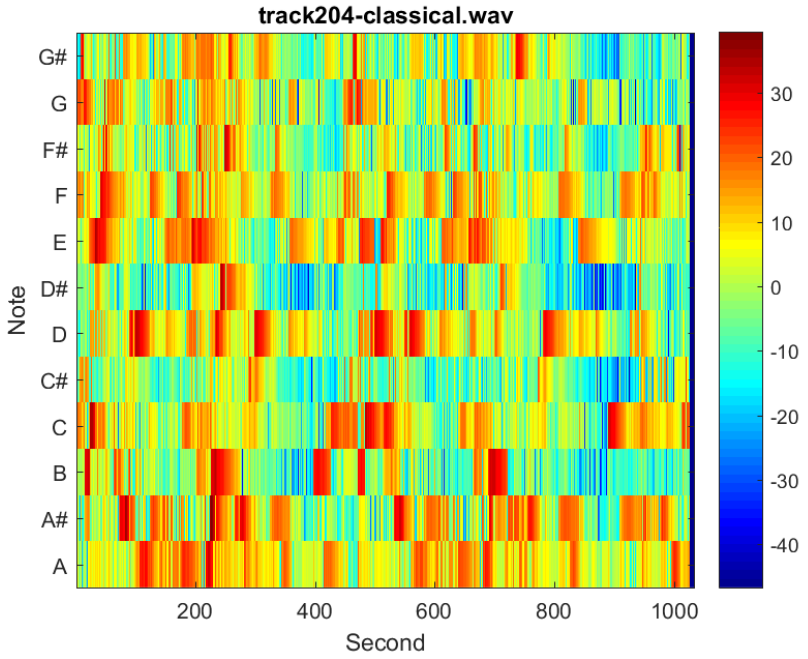


Fig: 50 NPCP for track 204

The song Schumann Davidsbundlertanze is simpler to follow than some other examples. The melody from the piano can be followed on the graph as the song's first phrase starts on an A#, rises to a G, then falls back down. So this track has a pattern of starting with low note, go up to high note and then go back down to low note. While I was listening to both of the classical music, I notice there aren't a lot of sharp and flat being play, it is mostly natural notes.

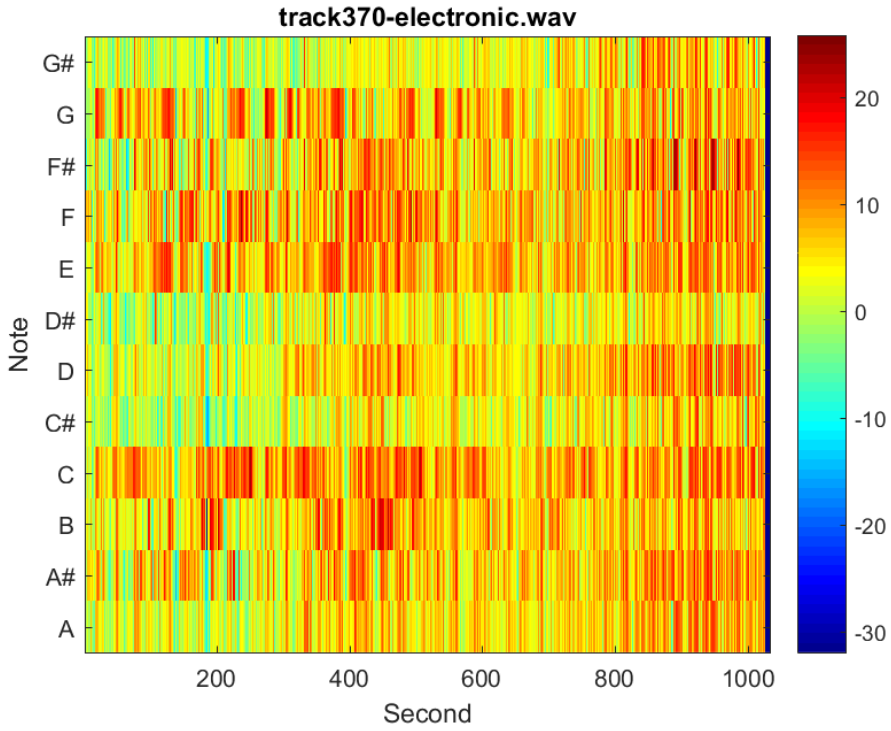


Fig 51: NPCP for track 370

The track 370 is an electronic song and electronic are made with computers, synthesizers and the drum machines and it makes it harder to distinguish from the sound. The song is very rhythmic. From the graph it starts with almost all the notes but more heavily on A,D,E,G because according to the graph it's more heavy marked in red. In the center the notes are rapidly rising and falling, because of the notes are changing rapidly. None of the note has been held for a long time.

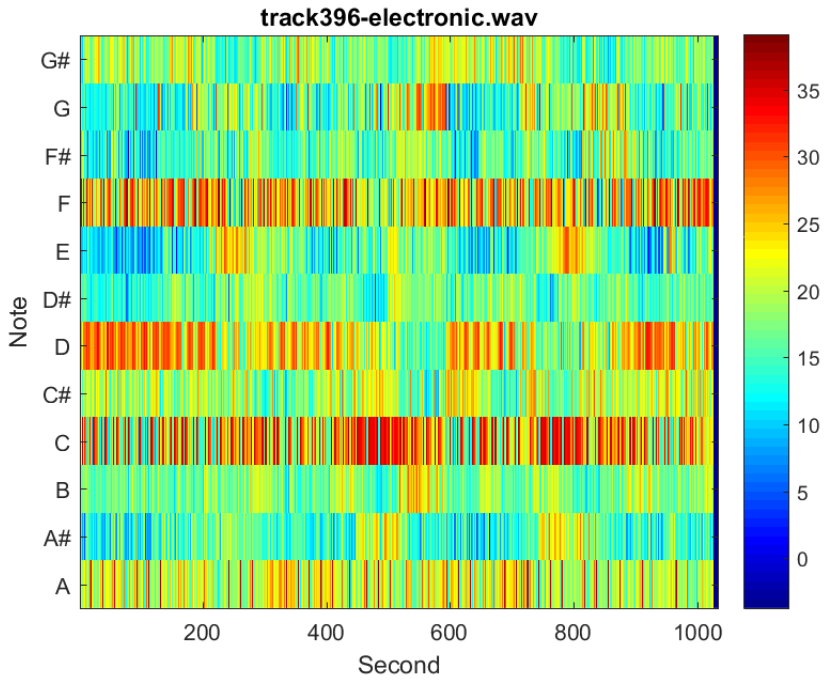


Fig 52: NPCP for track 396

The song track 396 can be classify as Trance Electronic. The Trance more like the classical form of electronic music because Trance style are more tonal but repetitive beat and melodic aesthetics as I can see from the graph the notes are repetitive. I can hear that they are typically faster beats and there is point in graph which it rest between drops, and then it adds vocal into the Trance electronic. So if you can see very fast switch between notes, it's probably electronic music.

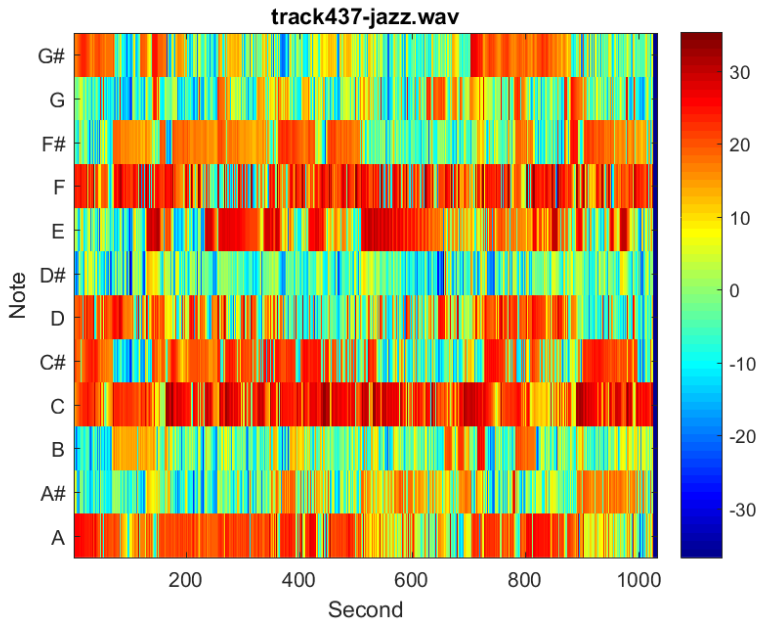


Fig 53: NPCP for track 437

The song is classify as fusion jazz, it can be a combination of two or more genres. It can be mixing of funk and rhythm and blues rhythms. In the beginning of music it's kind of quiet because the piano was starting off with C, D#, and rise to G notes. Then drums comes in and play which you see more decibel on the graphs. The seconds between 100 to 150 there were a rest going on and then both the piano and the drums starts off again that why you see the repetitive of same decibel happen around 50-100 and 130-230. Then piano finished off in the background toward the very end. The fusion jazz can be simple melodic that can be elaborates on the chord progressions.

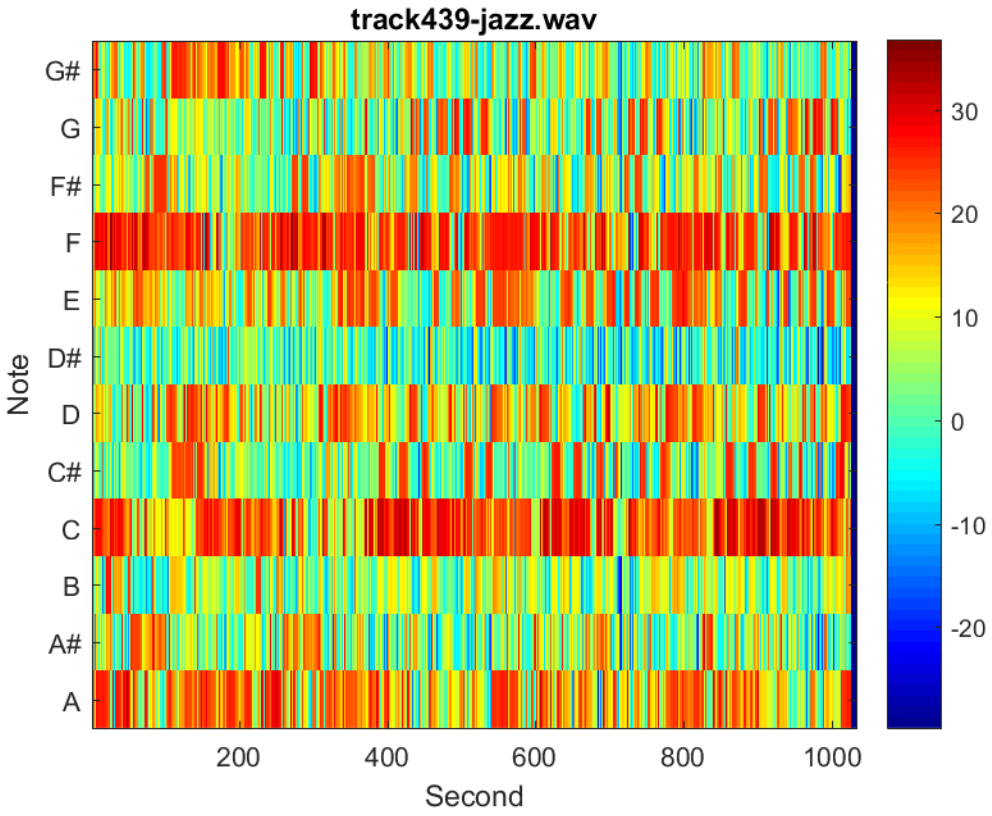


Fig 54: NPCP for track 439

The song wallawalla from Drop Trio starts of heavy with all the instrument at the beginning that's why the graph shows decibel of red and orange tone. During the 20-100 syncopation occurs which you can see in the graph, the decibel are shaded more heavy red, this is the place where it shifts the emphasis of song's rhythm and beat pattern. Then it transition back to normal rhythm and beat pattern. From 120-140 there is a transition where the keyboard have its own solo parts that why you see more heavy in A#, C, D#, F, G chords. Then it change back to same repetitive rhythm and beat pattern that why the graph show that 160-180 second is similar to the beginning parts.

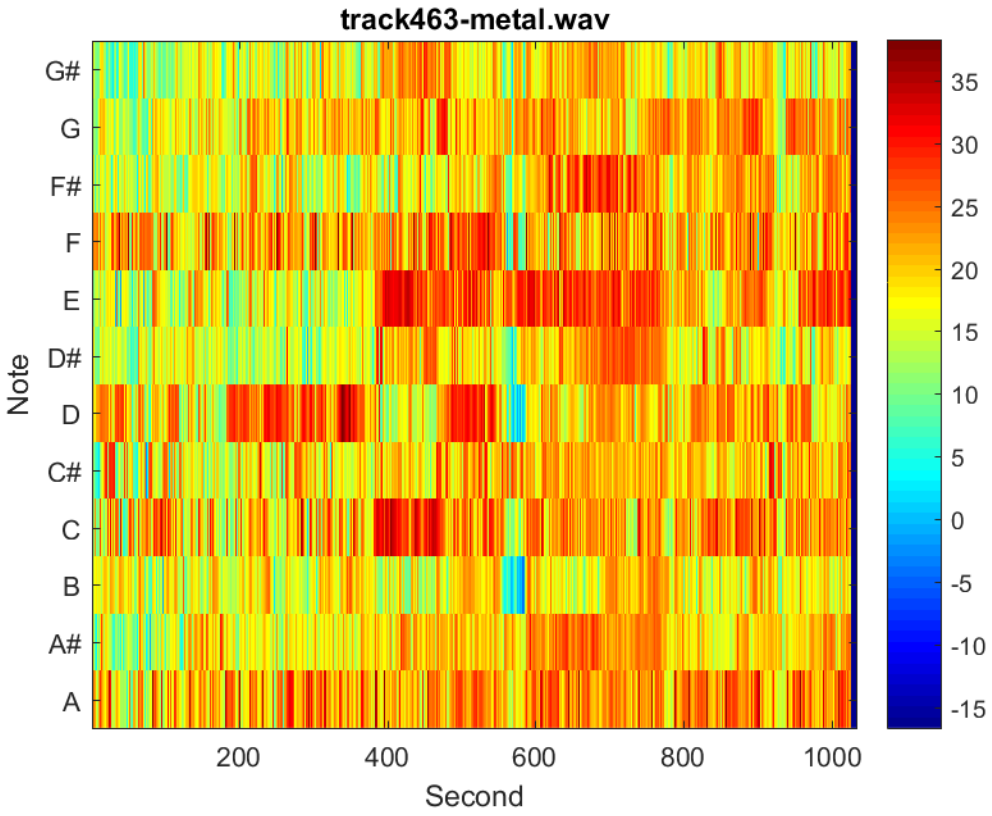


Fig 55: NPCP for track 463

The tempo of this track is extremely fast blast beat, therefore it also make it seems like you are hitting all of the notes at once, however, you are just hitting short two-note or three-note rhythmic. With the two-note and three-note rhythmic, you can see above the graph is having a lot of hits,so this can be classify as metal.

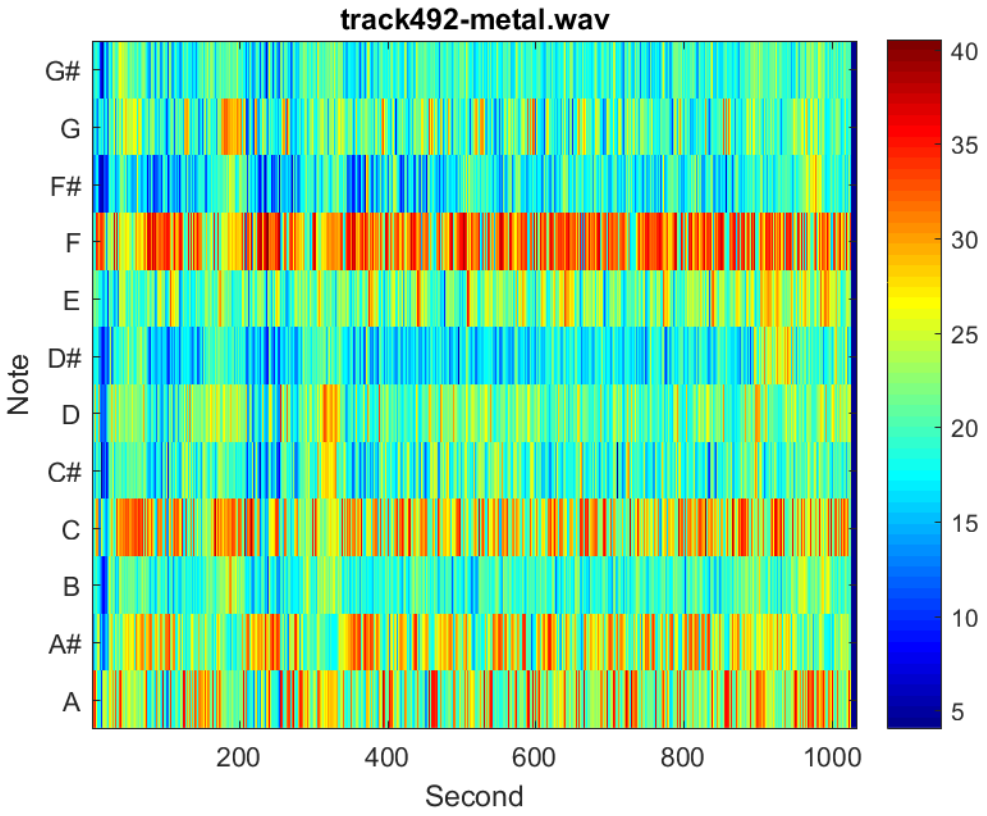


Fig 56: NPCP for track 492

When I am listening to track 492, you can hear the constant electric guitar as a base. The most common note for a electric guitar is tuned E,A,D,G and B. As you can see from the graph above, note A,D,E and G are the most common note for track 492. In this song, you can see the main groove is characterized by short, two-note or three-note rhythmic figures, usually you can classify this as metal.

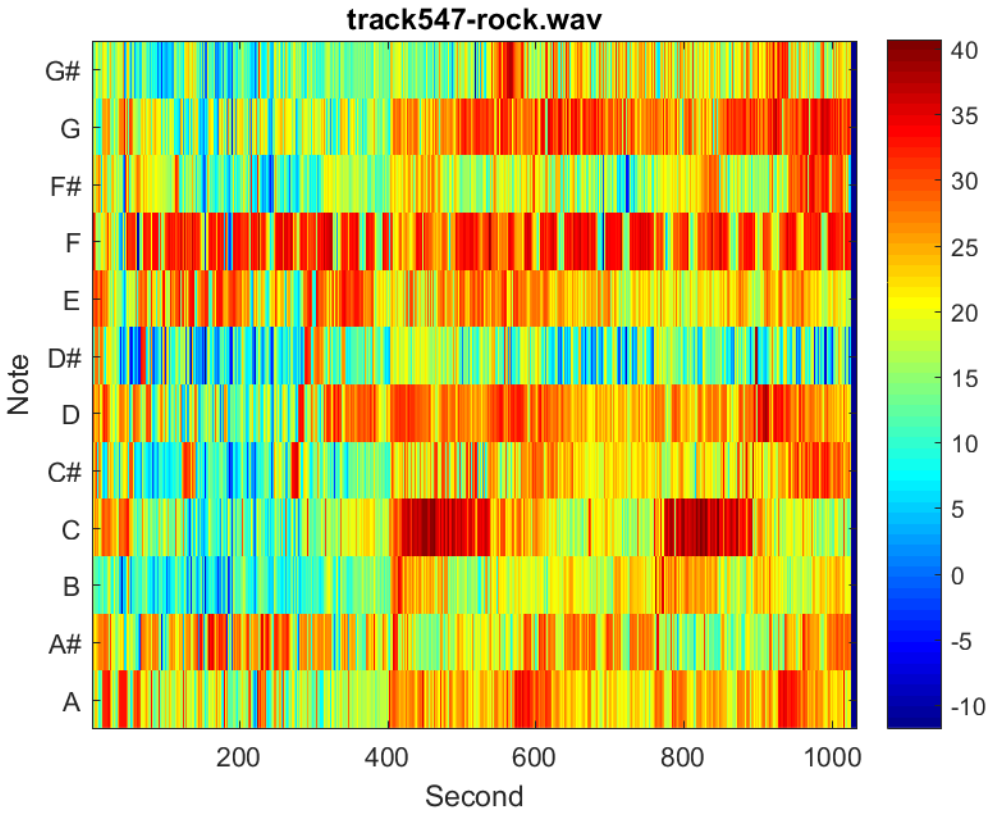


Fig 57: NPCP for track 547

This track 547 rock has the same electric guitar as the base, however, it doesn't have a short two-note or three-note rhythmic, so it looks really similar to metal, but it's not as note wasn't hitting in a fast blast beat tempo. So this song can be classify as rock.

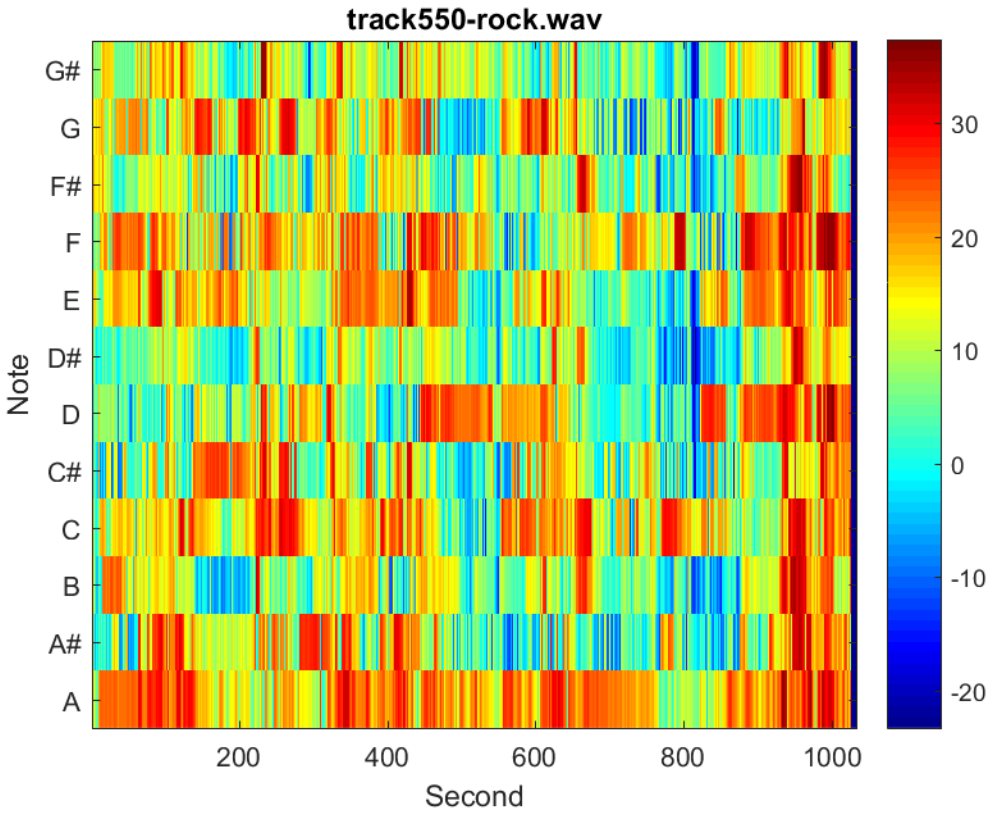


Fig 58: MPCP for track 550

In this rock song the singer is playing with rhythm section instrument such as guitar or bass. At the beginning of graph the bass starts off softly so the decibel is quieter. Later the vocal was added and played with bass. From 50-100 second all the instrument was playing along with vocal. From 100-140 the singer is playing rhythm with guitar and then it repeat back to playing together. The last part of the graph it also portrayed that the singer is playing along with guitar while bass is at the background playing simple notes.

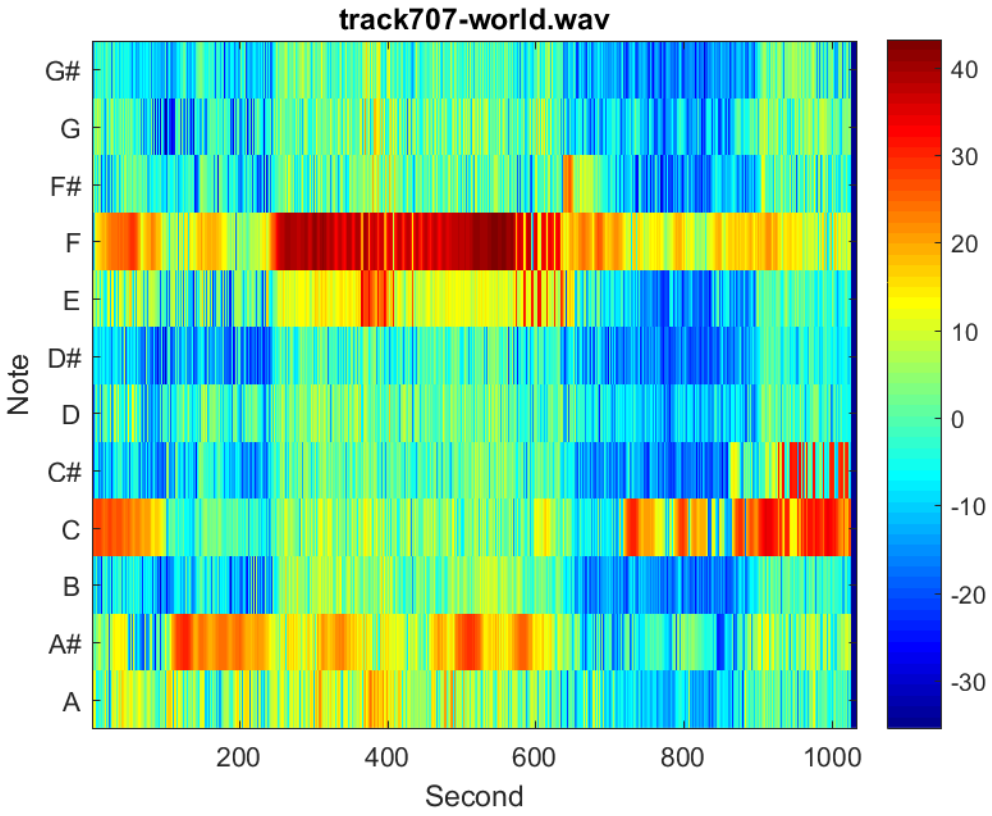


Fig 59: NPCP for track 707

At the beginning of the track 707, it's very quiet. You can see that on the graph above, and then you can hear the flute play D note throughout the whole song, since world music usually have either a relatively quiet start or end, in addition to that, it has been play in a monotonic tone. So this can be classify as world music.

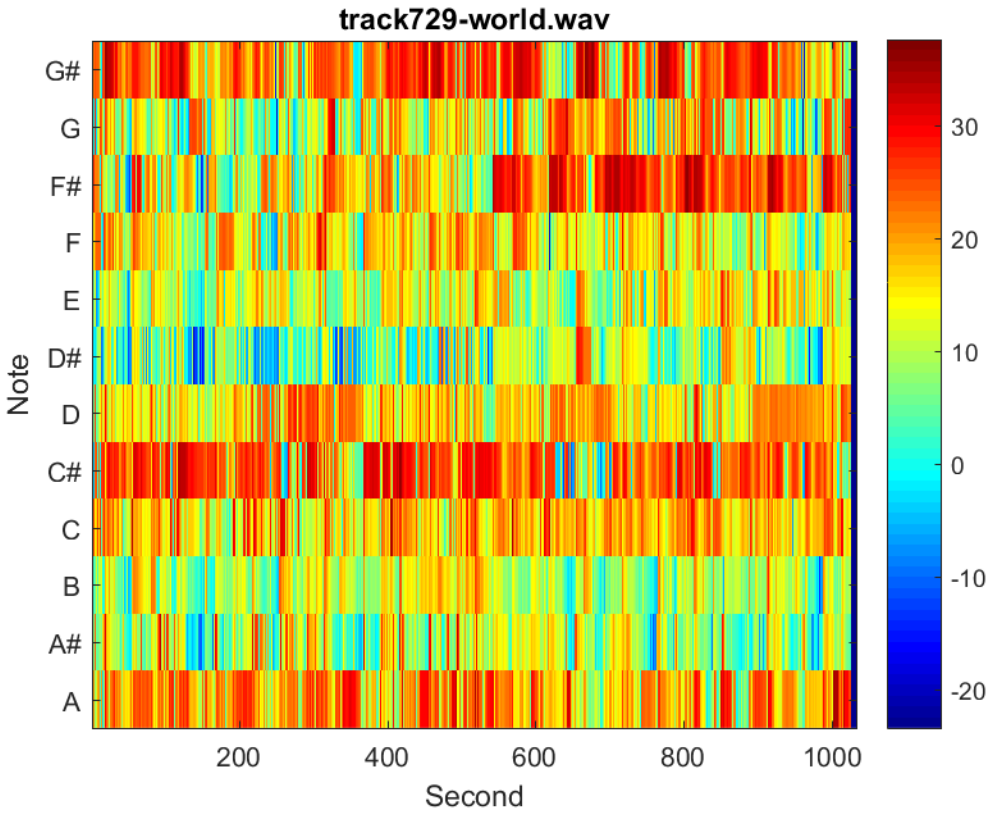


Fig 60: NPCP for track 729

This song may connect with traditional Japanese instruments such as the Koto and Shamisen, they are under the category of stringed instruments. The main notes are C,C#,F,G they are played throughout the whole song as indicated on the graph. The song was very consistent with the rhythm and repetitive. So it can be classify as world music.

Appendix:

Lab2.m

```
%This is Lab 2 of Digital Signal Processing
clear all;close all;clc;
tic
%------------Read in all the filename-------------------------
filename = {'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'};
for fileIndex = 1:12
    [song,fs] = audioread(char(filename(fileIndex)));
    fftSize = 512;
    kw = kaiser(fftSize);
% %     [song,fs] = audioread('track396-electronic.wav');
    ExtractedSong = (SongExtract(song,24,fs));
    nf = floor(24*fs/(fftSize/2));
    mfcc1 = zeros(40,nf);
    index = 1;
    for n = 1:256:(nf-1)*256
        mfcc1(:,index) = mfcc(ExtractedSong(n:n+fftSize-
1),fs,fftSize,kw);
        index = index + 1;
    end

    mfcc1 = 10*log10(mfcc1);
    sim = zeros(nf,nf);
    % 1. Implement the computation of the similarity matrix, and
display the similarity matrices (color-coded)
    % for the 12 audio tracks supplied for the _rst lab. See Fig. 1 for
an example of a similarity matrix.
%     for i = 1:nf
%         for j = 1:nf
%             sim(i,j) = SimilarityMatrix(mfcc1,i,j);
%         end
%     end
%     figure
%     imagesc(sim)
%     title(filename(fileIndex));
%     colormap jet
%     colorbar
%     saveas(gcf,strcat('Similarity',int2str(fileIndex),'.png'));
    %----------------------end of problem 1-----------------------------
----


    %Implement the computation of the rhythm index B(l) de_ned in (4).
Plot the vector B as a function
    % of the lag l = 0;Nf -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.
%     sumT = 0;
%     B = zeros(1,nf);
```

```
%     for l = 1:nf
%         for n = 1:nf-l
%             sumT = sumT + sim(n,n+l-1);
%         end
%         B(l) = 1/(nf-l-1)*(sumT);
%         sumT = 0;
%     end
%     figure
%     plot(B);
%     title(strcat('First Estimate for',{' '},filename(fileIndex)));
%     xlabel('lag(secs)')
%     %gcf stands for get current figure
%     saveas(gcf,strcat('FirstEst',int2str(fileIndex),'.png'));

    %--------------------End of problem 2-----------------------------
------
    % 3. Implement the computation of the rhythm index AR(l) de_ned in
(6). Plot the vector AR as a function
    % of the lag l = 0;Nf -1 for the 12 tracks. Comment on the
presence, or absence, of a strong rhythmic
    % pattern.
    % See Fig. 2 for an example of a plot of the rhythm index.
%     AR = zeros(1,nf-3);
%     for l = 1:nf-3
%         for i = 1:nf-1
%             for j = 1:nf-2-l
%                 product(i,j) = sim(i,j)*sim(i,j+l-1);
%             end
%         end
%         AR(l) = 1/(nf*(nf-l))*sum(sum(product(:,:)));
%         product(:,:) = 0;
%     end
%     figure
%     plot(AR)
%     title(strcat('A better Estimate for',{' '},filename(fileIndex)));
%     xlabel('lag(sec)')
%     saveas(gcf,strcat('BetterEst',int2str(fileIndex),'.png'));

    % 4. Implement the computation of the rhythm index AR(l;m) de_ned
in (8). Plot the image AR in false
    % color as a function of the lag l = 0; : : : ; 19 (y-axis) and the
time window index m (x-axis) for the 12
    % tracks. Comment on the variation of rhythm patterns for the
di_erent tracks.

%     ShortSong = (SongExtract(song,1,fs));
%     nfs = floor(1*fs/(fftSize/2));
%     mfcc2 = zeros(40,nfs);
%     index = 1;
%     for n = 1:256:(nfs-1)*256
%         mfcc2(:,index) = mfcc(ShortSong(n:n+fftSize-
1),fs,fftSize,kw);
%         index = index + 1;
%     end
%
%     Sim1sec = zeros(nfs,nfs);
```

```
%
%     for i = 1:nfs
%         for j = 1:nfs
%             Sim1sec(i,j) = SimilarityMatrix(mfcc2,i,j);
%         end
%     end
% %     figure
% %     imagesc(Sim1sec)
% %     title('Similarity Matrix for 1 seconds')
% %     colormap jet
% %     colorbar
%
%     prod = zeros(20,20);
%     for l = 1:20
%         for m = 1:floor(nfs/20)-1
%             for i = 1:20
%                 for j = 1:20-l+1
%                     prod(i,j) =
Sim1sec(i+m*20,j+m*20)*Sim1sec(i+m*20,j+m*20+l-1);
%                 end
%             end
%             ARlm(m,l) = 1/(20*(20-l))*sum(sum(prod(:,:)));
%             prod = 0;
%         end
%     end
%     figure
%     imagesc(ARlm);
%     title(strcat('AR on the variation of rhythm patterns for',{'
'},filename(fileIndex)));
%     ylabel('lag(s)')
%     xlabel('times(s)')
%     color map
%     saveas(gcf,strcat('AR',int2str(fileIndex),'.png'));

    %-------------------------End of problem 4-------------------------
---
    % 5. Implement the computation of the Normalized Pitch Class
Pro_le, de_ned by (18). You will compute
    % a vector of 12 entries for each frame n.
    % 6. Evaluate and plot the NPCP for the 12 audio tracks supplied
for the _rst lab.
    % See Fig. 4 for an example

    Threshold = max(max(song));
    index = 1;
    %Allocate a 12 by number of frames vector
    output = zeros(12,nf);
    %Passing each frame of the song into the NPCP funciotn
    for n = 1:fftSize/2:floor(length(ExtractedSong)-2048)
        output(:,index) = NPCP(ExtractedSong(n:n+fftSize-
1),fs,fftSize,kw,Threshold);
        index = index + 1;
    end
    output = 10*log10(output/Threshold);
    figure
    imagesc(output);
```

```
    title(filename(fileIndex));
    set(gca,'YDir','normal');
    set(gca,'YTick',[1:12]);

set(gca,'YTickLabel',({'A';'A#';'B';'C';'C#';'D';'D#';'E';'F';'F#';'G';
'G#'}))
    xlabel('Second');
    ylabel('Note');
    colormap jet
    colorbar
    saveas(gcf,strcat('NPCP for',int2str(fileIndex),'.png'));
end
toc
```

Mfcc.m

```
function [ output ] = mfcc( wav,fs,fftSize, window )
% MFCC(Mel-frequency cepstral coefficients)
%
%USAGE
% [mfcc] = mfcc(wav, fs, fftSize,window)
%
%INPUT
% vector of wav samples
% fs: sampling frequency
% fftSize:size of fft
% window: a window of size fftSize
%
%OUTPUT
% mfcc (matrix size) coefficients x nFrames
% harewired parameters
hopSize = fftSize/2;
nBanks = 40;
% minimum and maximum frequencies for the analysis
fMin = 20;
fMax = fs/2;
%_____________________________________________________________________
%
% PART 1 : construction of the filters in the frequency domain
%_____________________________________________________________________
% generate the linear frequency scale of equally spaced frequencies
from 0 to fs/2.
linearFreq = linspace(0,fs/2,hopSize+1);
fRange = fMin:fMax;
% map the linear frequency scale of equally spaced frequencies from 0
to fs/2
% to an unequally spaced mel scale.
melRange = log(1+fRange/700)*1127.01048;
% The goal of the next coming lines is to resample the mel scale to
create uniformly
% spaced mel frequency bins, and then map this equally spaced mel scale
to the linear
%& scale.
% divide the mel frequency range in equal bins
melEqui = linspace (1,max(melRange),nBanks+2);
fIndex = zeros(nBanks+2,1);
% for each mel frequency on the equally spaces grid, find the closest
frequency on the
% unequally spaced mel scale
for i=1:nBanks+2
[dummy,fIndex(i)] = min(abs(melRange - melEqui(i)));
end
% Now, we have the indices of the equally-spaced mel scale that match
the unequally-spaced
% mel grid. These indices match the linear frequency, so we can assign
a linear frequency
% for each equally-spaced mel frequency
fEquiMel = fRange(fIndex);
% Finally, we design of the hat filters. We build two arrays that
correspond to the center,
% left and right ends of each triangle.
```

```
fLeft = fEquiMel(1:nBanks);
fCentre = fEquiMel(2:nBanks+1);
fRight = fEquiMel(3:nBanks+2);
% clip filters that leak beyond the Nyquist frequency
[dummy,tmp.idx] = max(find(fCentre <= fs/2));
nBanks = min(tmp.idx,nBanks);
% this array contains the frequency response of the nBanks hat filters.
freqResponse = zeros(nBanks,fftSize/2+1);
hatHeight = 2./(fRight-fLeft);
% for each filter, we build the left and right edge of the hat.
for i=1:nBanks
freqResponse(i,:) = ...
(linearFreq > fLeft(i) & linearFreq <= fCentre(i)).* ...
hatHeight(i).*(linearFreq-fLeft(i))/(fCentre(i)-fLeft(i)) + ...
(linearFreq > fCentre(i) & linearFreq < fRight(i)).* ...
hatHeight(i).*(fRight(i)-linearFreq)/(fRight(i)-fCentre(i));
end
%
% plot a pretty figure of the frequency response of the filters.
% figure;set(gca,'fontsize',14);
% semilogx(linearFreq,freqResponse');
% axis([0 fRight(nBanks) 0 max(freqResponse(:))]);title('FilterbankS');
%_____________________________________________________________
%
% PART 2 : processing of the audio vector In the Fourier domain.
%_____________________________________________________________
%
% YOU NEED TO ADD YOUR CODE HERE

    Y = fft(wav.*window);
    K = fftSize/2+1;
    Xn = Y(1:K);
    output = (abs(freqResponse*Xn)).^2;
end
```

NPCP.m

```
function [ output ] = NPCP( wav,fs,fftSize,window,Threshold)
        %
        % PART 2 : processing of the audio vector In the Fourier
domain.
        %_____________________________________________________________
        %
        Y = fft(wav.*window);
        K = fftSize/2+1;
        % L = K-1;
        Xn = abs(Y(1:K));
        %-------------Plot of the fourier transform--------------------
------
        % f = fs*(0:L)/L;
        % plot(f,Xn);
        %--------Step 1 pick out the peak frequencies-------
        f0 = 27.5;
        %------Goal of the threshold:Get rid of the small amplitude
note----
        [pkt,fk] = findpeaks(Xn,'Threshold',Threshold*1e-6);
        % [pkt,fk] = findpeaks(Xn,'NPeaks',20);
        FK = round(fk*fs/fftSize);
        %---------Step 2: Assignment of peak frequencies to
        %------------------------------semitones-----------------------
-
        sm = round(12*log2(FK/f0));
        c = mod(sm,12)+1;
        r = 12*log2(FK/f0)-sm;
        output = zeros(12,1);
        for j = 1:12
            var = ismember(c,j);
            w = (cos(pi/2*r)).^2;
            output(j,1) = sum(var.*(w.*pkt.^2));
        end
    end
```

SimilarityMatrix.m

```
function [output] = SimilarityMatrix(mfcc,i,j)

    output = mfcc(:,i)'*mfcc(:,j)/(norm(mfcc(:,i))*norm(mfcc(:,j)));

end
```

SongExtract.m

```
function [output] = SongExtract(song,NumSec,fs)
SongLength = length(song);
GrabLength = NumSec*fs;
Mid = floor(SongLength/2);
    if SongLength <= GrabLength
        output = song;
    else
        output = song(Mid-floor(GrabLength/2):Mid+floor(GrabLength/2)-
1);
    end
end
```