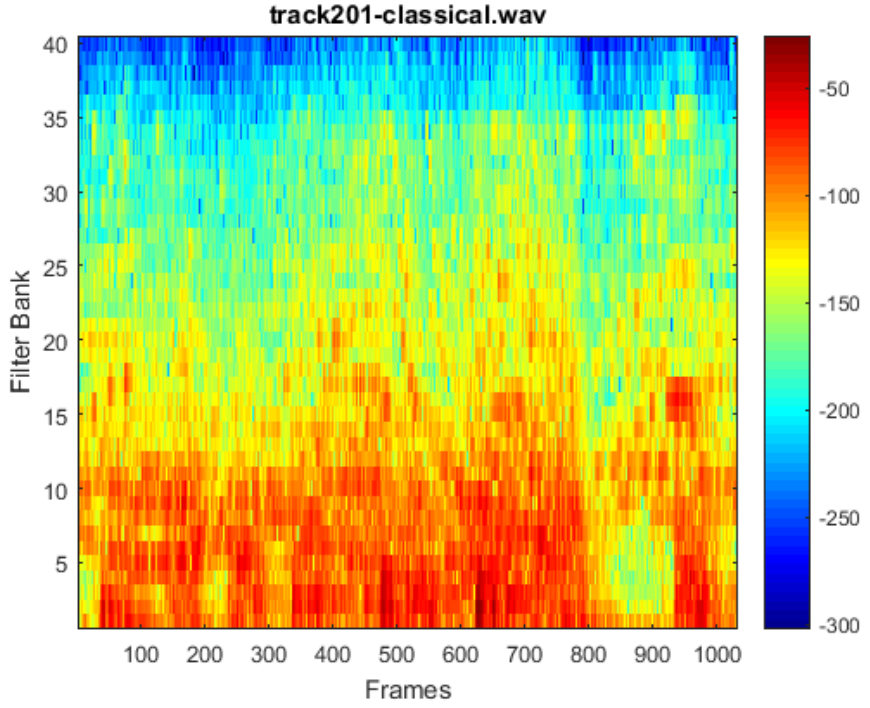
Final project part I

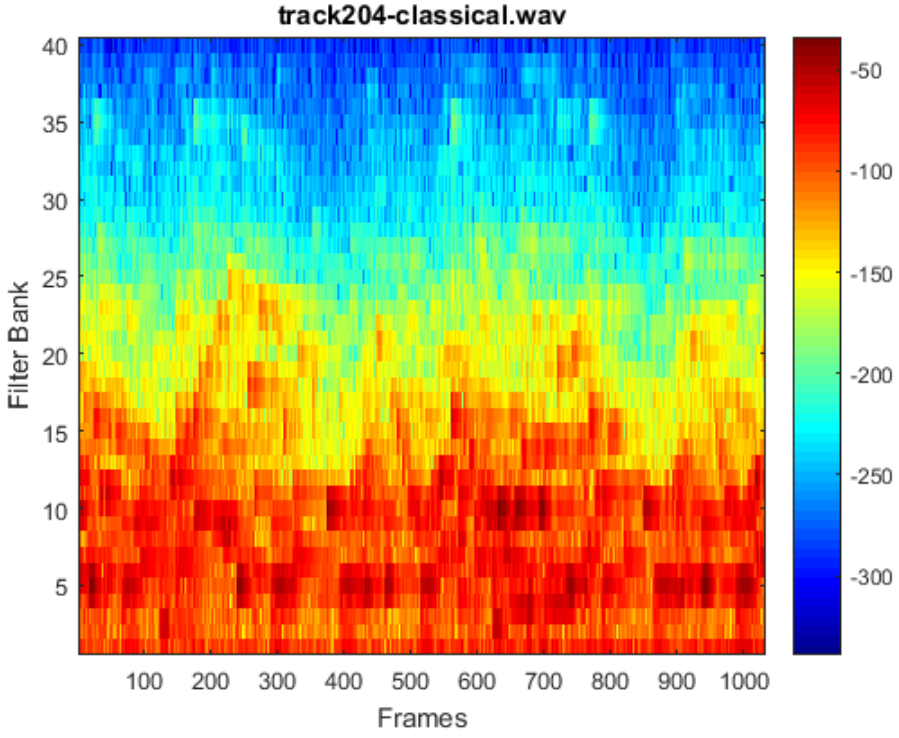
Zhi Jie Huang

ECEN 4632

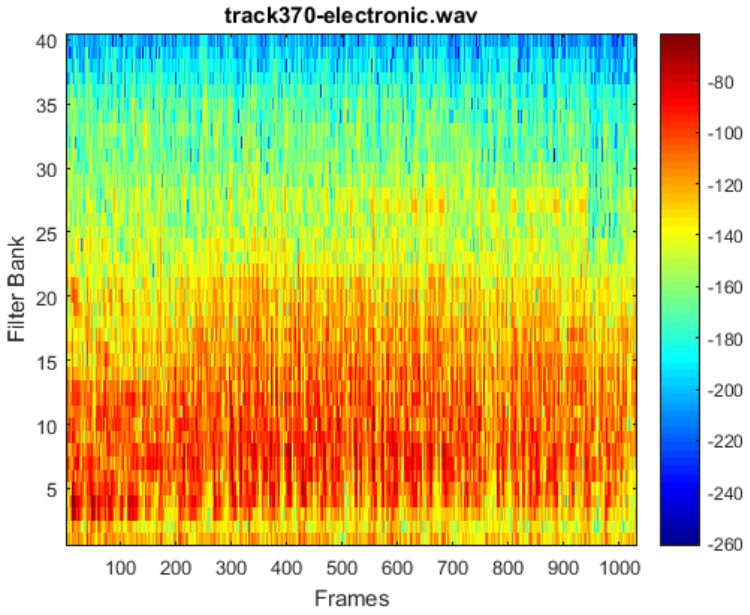
Project partner: Richard Smith

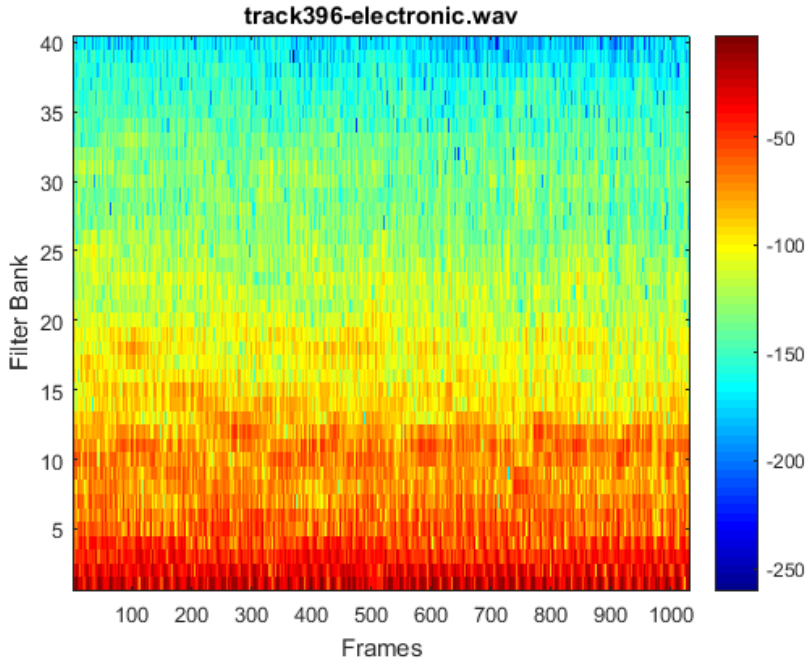
1. I didn't create a function to extract my T seconds of the music, instead I did it in a main script name FinalProjectPartI.m. The theory is behind the extraction is since you will get a complete different melody and rhythm, etc. in the middle of the song vs the beginning of the song. Therefore, it's suggested take the 24 seconds of music from middle of the given track. So you calculated the middle of the song by taking the floor of the length of the song/2. Since the sampling frequency fs is 11025, when you take the 24 seconds of the song. You need to multiply by the sampling frequency in order to successfully extract 24 seconds.
2. The way to calculate the mfcc(Mel-frequency cepstral coefficients) is to pass in 512 samples at a time into my mfcc.m, in addition to that, you are moving the frame of your song at half of the frame which it's 256 samples at a time. Because you want your signal to overlapped in order to prevent the loss of information. I am using the hann window to cookie-cut my signal, but you can also use Kaiser window to truncate your signal as well. In my mfcc.m, I did a Fast Fourier transform on the 512 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, you truncate the negative frequency of the fft signal by taking the first signal up to its nyquist frequency (i.e. 512/2+1 = 257 in this case). Since the Hp is already defined in the mfcc.m, all I have to do to get the mel-spectrum(MFCC) coefficient is to sum up the multiply my fft signal with the 40 continuous cochlear filterbank, do an absolute value on the sum and then square the result. Doing a matrix multiplication in matlab will take care of the sum and multiplication for you. Using the filterbank gives us a way to calculated the total energy at the output of each filter bank.
3. Output of all 12 audio tracks:
    a. The output of the two classical looks very like each other, since classical music usually consisted pf flutes, oboes, horn and violins. They usually are high frequency and lower amplitude.


track201-classical.wav
Filter Bank
Frames
40
35
30
25
20
15
10
5
100
200
300
400
500
600
700
800
900
1000
-50
-100
-150
-200
-250
-300

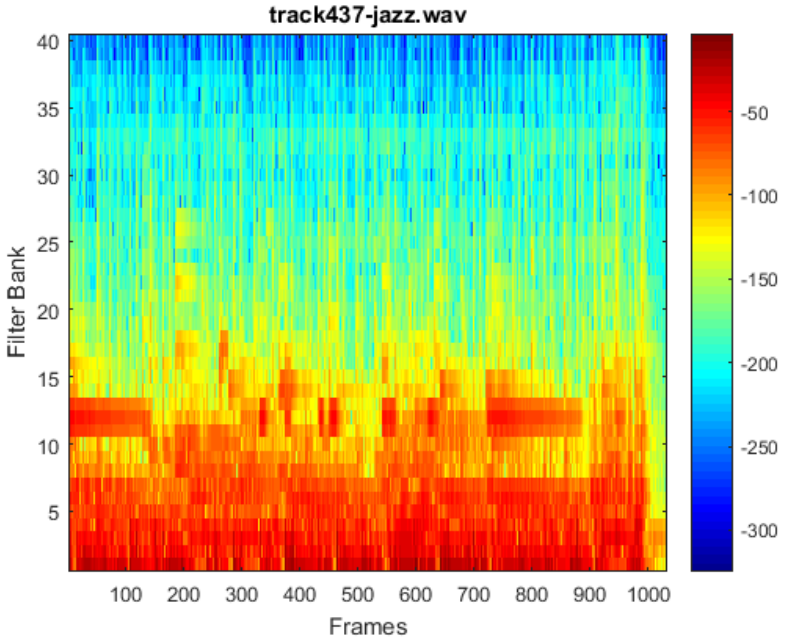

track204-classical.wav
Filter Bank
Frames
40
35
30
25
20
15
10
5
100
200
300
400
500
600
700
800
900
1000
-50
-100
-150
-200
-250
-300

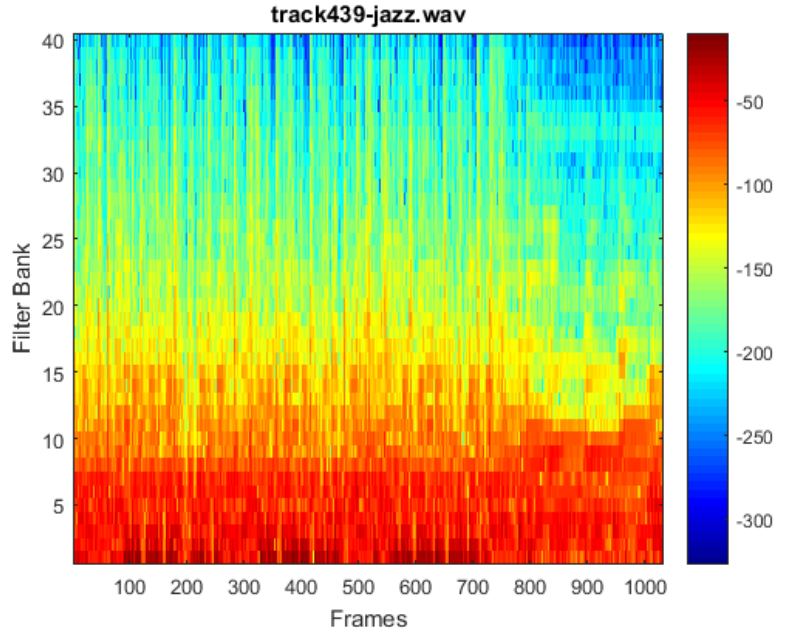
Usually Electronic music have a lot of bass, therefore, it’s low frequency.



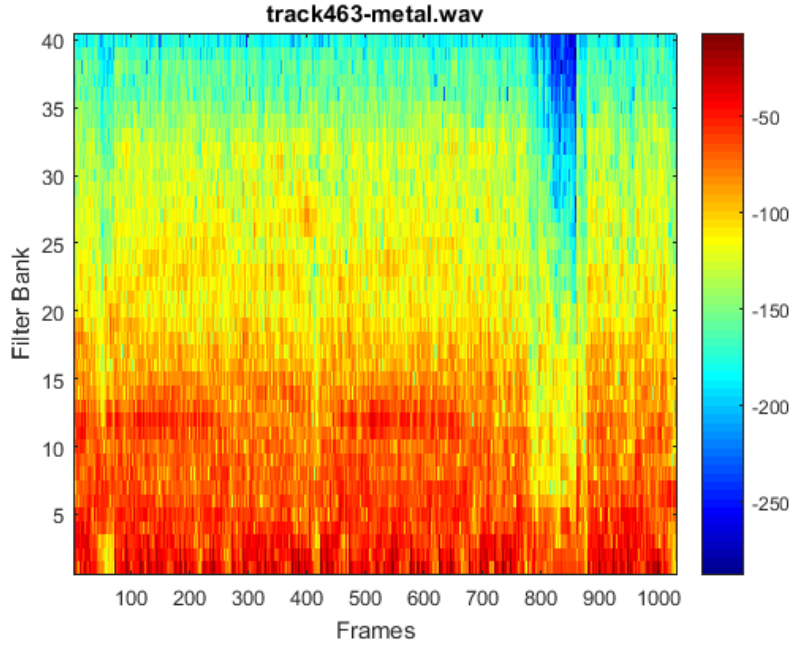


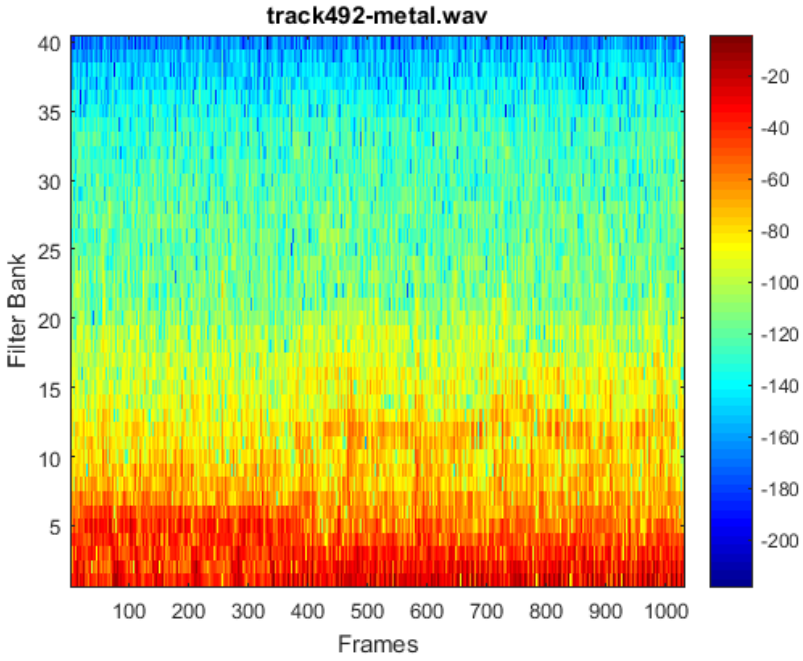
Usually jazz can be played on saxophone, trumpet, trombone. Saxophone is high frequency and low amplitude, so it matches up with the frequency response below.



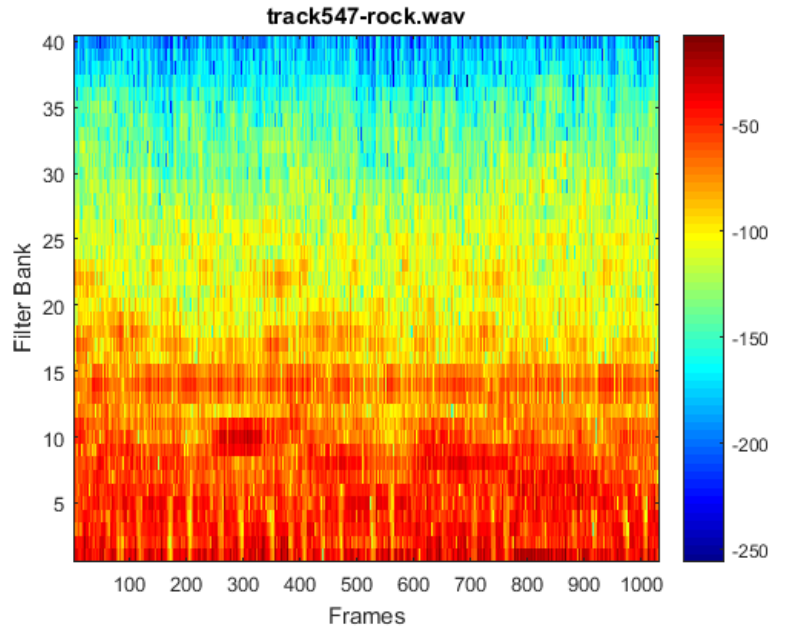


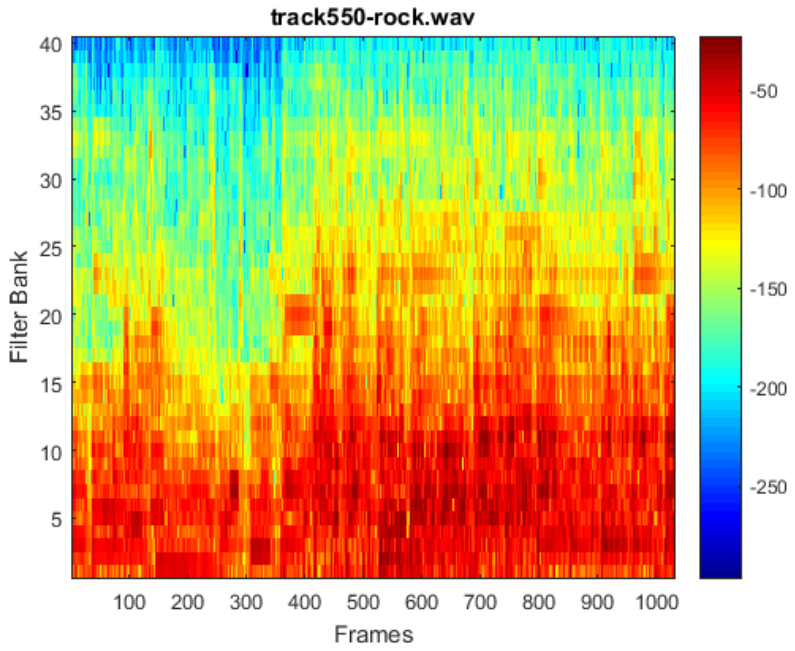
Metal music is high frequency and low intensity.



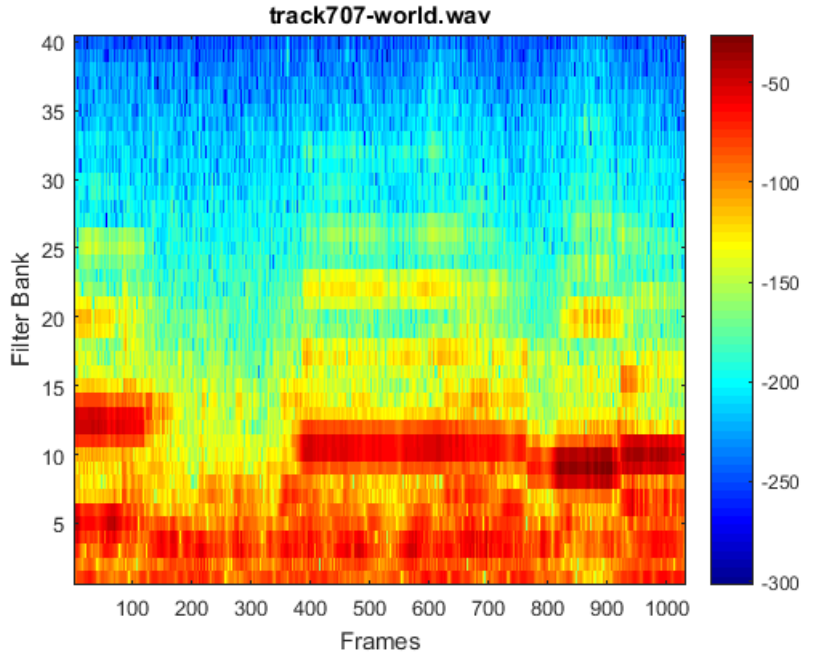


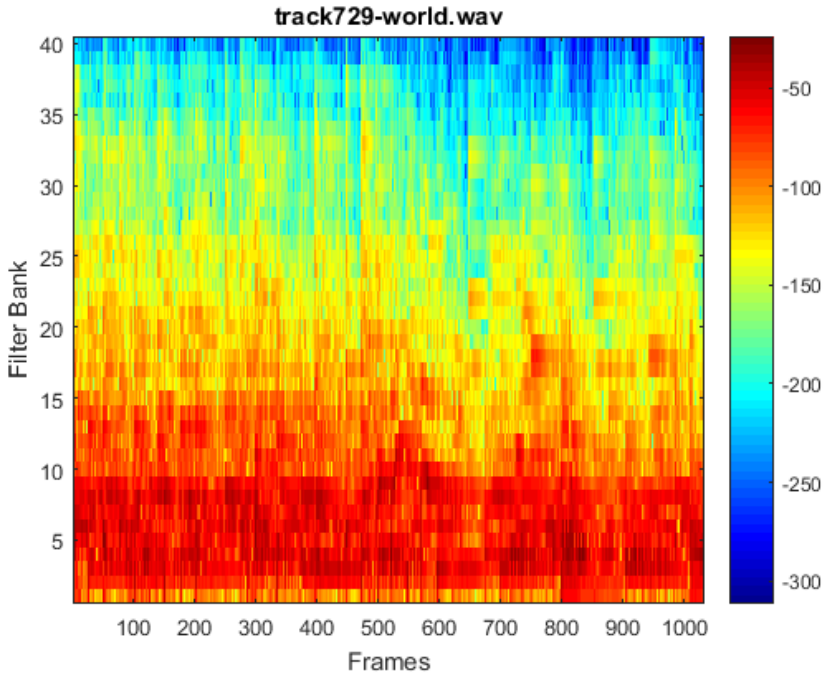
In general, rock music consists of electrical guitar, bass and drum. They are all high frequency and low amplitude.





Since world music consists of variety of instruments, therefore, it will have some high frequency, high amplitude, or low frequency and low amplitude. Therefore, the frequency response is about right.





Appendix:

FinalProjectPartI.m

```
clear all;close all;clc;
%%%%%%%%%%%%%Final project part I From the wav file to the psychoacoustic
%%%%%%%%%%%%%features
%%--------------use dir **/*.wav to list all the wav file in the directory
%tic
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)));
    %%%% audio files are sampled at fs = 11025 Hz
    %sound(song,fs)

    %%%%%%--------------initialization----------------------
    SongLength = length(song);
    Mid = floor(SongLength/2);
    %%%%-----------------------Part 1--------------------------------
    %% Extract 24 seconds of music from a given track.
    xn = song(Mid:Mid+24*fs-1);
    w = hann(512);
    fftsize = 512; %%%%Size of fft
    nf = floor(24*11025/256);
    index = 1;
    %%%%%% nbank = 40
    output = zeros(40,nf);
    for n = 1:256:(nf-1)*256
        output(:,index) = mfcc(xn(n:n+fftsize-1),fs,fftsize,w);
        index = index + 1;
    end

    %%%%%%%Take 20log10 of your output,
    %use flipud

    %output = flipud(output);
    output = 20*log10(output);

    figure
    imagesc(output);
    title(filename(fileIndex));
    set(gca,'YDir','normal');
    xlabel('Frames');
    ylabel('Filter Bank');
```

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