AMATH 482 Assignment 2:

Isolating an Instrument in an Audio File

Yuqing Cui
February 10, 2021

Abstract

In this report, we are provided with two audio files. The GNR clip is a 14-second file playing the song *Sweet Child O' Mine* by Guns N' Roses. The Floyd clip is a one-minute-long file playing *Comfortably Numb* by Pink Floyd. We utilized the Gabor transform to generate spectrograms that allows us to reproduce the music score of the guitar in the GNR clip and that of the bass in the Floyd clip. Then we isolated the bass and guitar solo in the Floyd clip by applying multiple filters in the frequency domain.

I. Introduction and Overview

In a music clip, there is always more than one instrument. It's very common for people to consider pulling one instrument out. We often deal with music in the time domain. With the Fourier transform, we are able to see what happens in the frequency domain as well. However, when we transform our data using the Fourier transform, we would lose the information about time that we would not be able to find out at what time a certain frequency occurs. The Gabor transform, on the other hand, allows us to preserve the information of the time and the frequency at the same time. The purpose of this report is to implement the Gabor transform to analyze the note played as time goes on by a specific instrument in the given clip.

II. Theoretical Background

In Assignment 1, we have introduced the Fast Fourier transform (FFT) $\hat{f}(k)$ and its inverse $f(x)$ as follows:

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

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

in which $e^{i\theta} = cos(\theta) + isin(\theta)$ by Euler's formula, $f(x)$ is a function of space or time, and $\hat{f}(k)$ is a function of frequency k. We can use FFT in MATLAB by the function `fftn`. However, we have to pay attention that FFT assumes a 2π periodic domain and we have to use `fftshift` to make sure that the transform is visualized properly. Though FFT allows us to look at data in the frequency domain, we lose all the information about time. In other words, we are unable to visualize the data in the time domain.

The Gabor transform, also called the short-time Fourier transform (STFT), solves the problem. It is given by:

$$\tilde{f}_g(\tau, k) = \int_{-\infty}^{\infty} f(t) g(t - \tau) e^{-ikt} dt, \tag{3}$$

in which $f(t)g(t-\tau)$ is a filter function centered at τ. We can see that it localized inputs over a chosen small window of time. Therefore, we can gain both time and frequency. We can choose any function $g(t)$ as long as it's real and symmetric and the L_2-norm of it is set to unity. Our preferred choice here would be a Gaussian filter in the following form:

$$F(k) = e^{-\tau(k-k_0)^2}, \tag{4}$$

where τ is the width of the filter that is manually determined and k_0 is the center frequency that we want to filter around. The Gabor transform also has drawbacks. Similar to the Heisenberg uncertainty principle, we cannot get perfect data in both the time domain and the frequency domain at the same time. As we gain more information in the time domain, we would lose more information in the frequency domain and vice versa.

III. Algorithm Implementation and Development

We started by loading the audio data. We set the time scale in seconds and prepared for the Gabor transform later by rescale the frequency k by a factor of $\frac{2\pi}{L}$ because it assumes 2π periodic signals.

Next, we began to build the Gabor filter. We decided to use a Gaussian filter in this assignment as the filter function. We choose 100 to be the filter width in the time domain as our basic filter and 0.001 to be that in the frequency domain to filter out overtones. We've also tried several other possible parameters. As long as the filter width in the time domain falls in the range $[100, 1000]$ and the filter width in the frequency domain falls in the range $[0.0001, 0.001]$, the output would be readable and acceptable. Other parameters beyond those range would create dispersed spectrograms.

We applied the Gabor filter to the data at each time step and then transformed them into the frequency domain. We still need to apply another filter because there are overtones in the clip. Since the guitar in the GNR clip and the bass in the Floyd clip are clearly identifiable, we can simply filter around the greatest frequency to eliminate overtones. We then plotted the spectrograms of the two clips after filtering and determined the music score by looking at the frequency.

Our next task is to isolate the bass in the Floyd clip. According to information online (www.zytrax.com), the frequency of a bass is from 40 Hz to 250 Hz. Thus, we set the maximum frequency that we would focus on to 250 Hz and amplified those frequencies by applying a Gaussian filter around the center frequency we got. We can get other instruments filtered out and the bass being reinforced.

Lastly, we are asked to pull the guitar solo out of the Floyd clip as clear as possible. It's challenging because the guitar is not very distinct with the bass, the drum, overtones, and other irrelevant instruments. The frequency of drums is usually less than 250 Hz, which is about the same as the bass. Therefore, we first filtered the bass and the drum (audio with a frequency less than 250 Hz) out of the frequency domain. We then applied the second filter to reinforce the frequency below 1200 Hz because the guitar usually produces such frequency. As a result, the frequency between 250 Hz and 1200 Hz should be clearer. However, it doesn't solve the issue brought by overtones falling in such regions. If we could find the center frequency of overtones,

creating a filter by summing them up and applying it to the data, the result might be more concentrated.

IV. Computational Results

All the plots below are log-frequency spectrogram (i.e. plotting $log(|s| + 1)$ instead of the original spectrogram s). We set it logarithmic to emphasize the tonal relationship.

(a) Music Score

Figure 1. is the spectrogram of the guitar in the GNR clip. I've labeled the music score corresponding to each frequency. The score follows Middle C, High D, High A, Middle G, High A, and so on. The three points representing High High D and High High C might be resulted from incomplete filtering about overones.

Figure 2. is the spectrogram of the bass in the Floyd clip, with the music score corresponding to each frequency labeled on the right. The score follows Low B, Low A, Low Low F, Low B, Low F, Low B, and so on. Since bass should produce frequency less than 250 Hz, those points above Middle E might be resulted from incomplete filtering about overones.

If we try different parameters for our filter, the spectrogram might be improved.

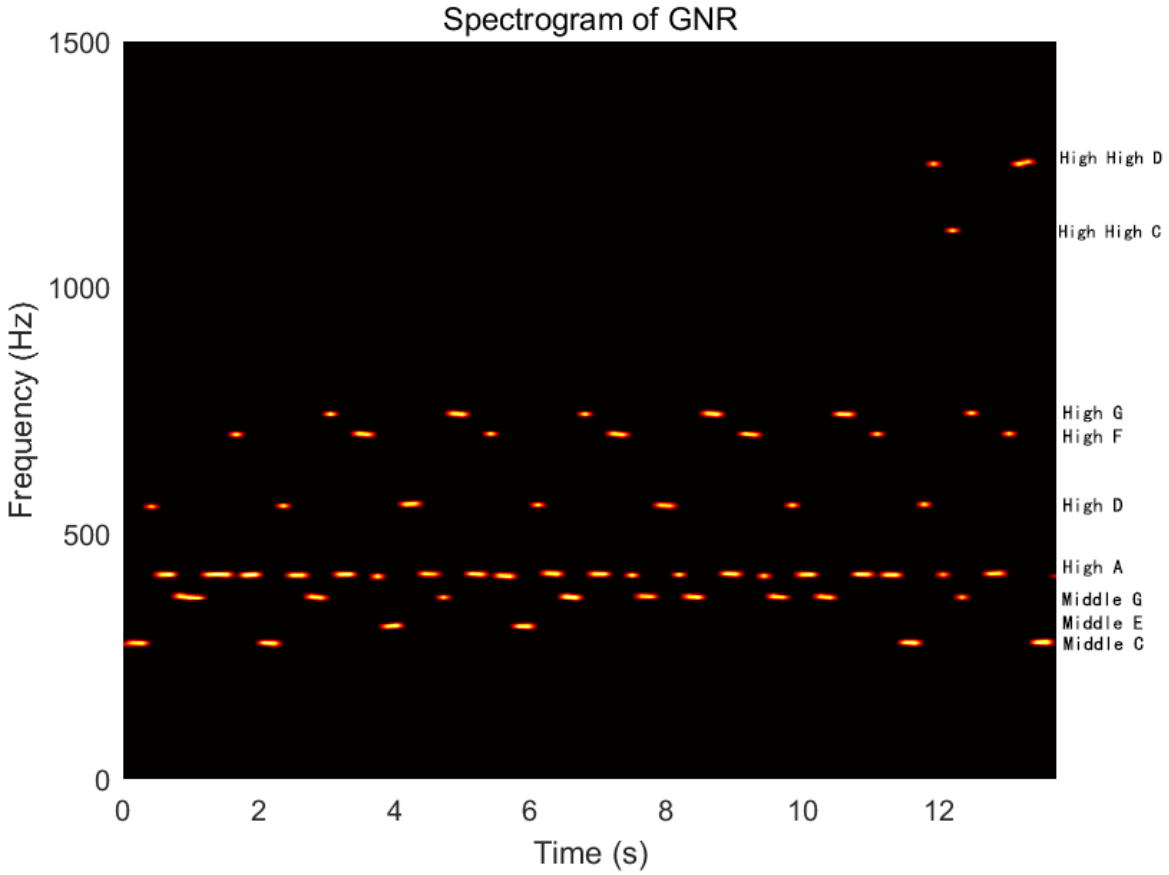

Figure 1. The spectrogram of the guitar in the GNR clip

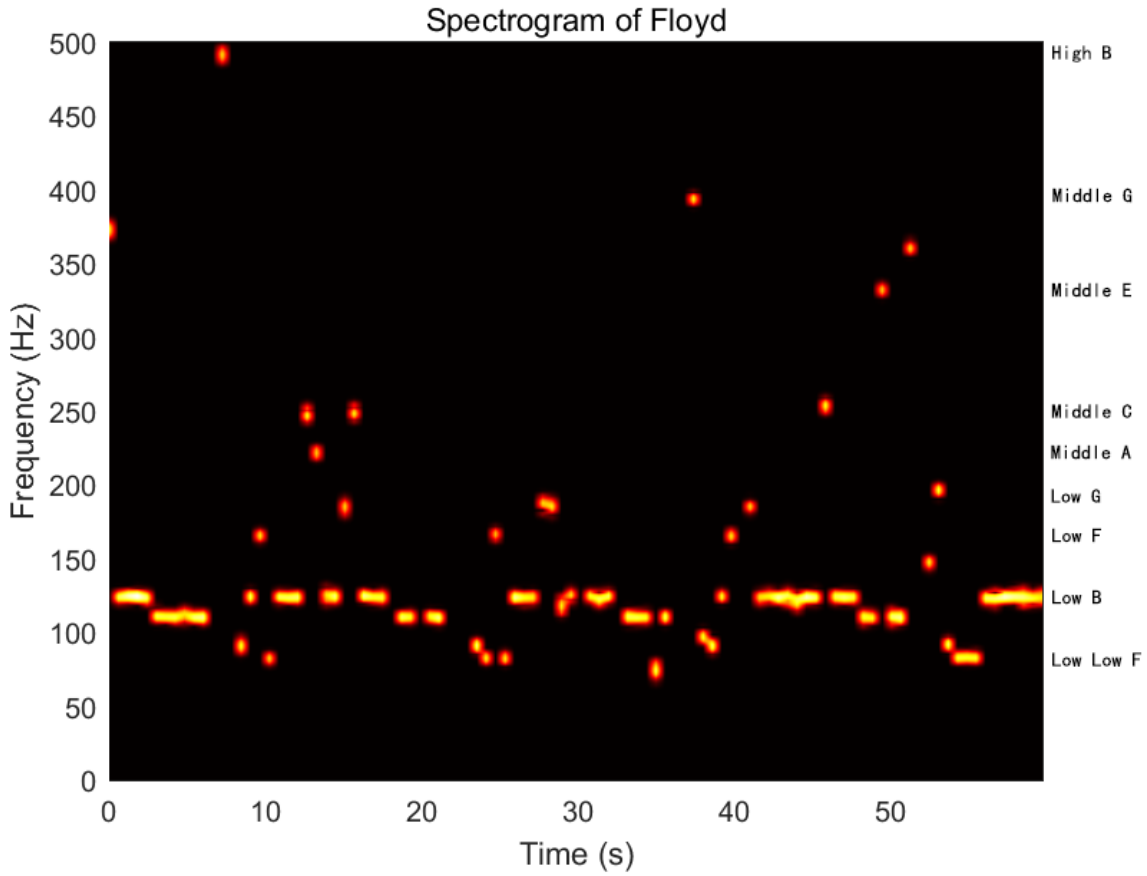

Figure 2. The spectrogram of the bass in the Floyd clip

(b) Isolation of the Bass

Figure 3. is the spectrogram of the bass in the Floyd clip after applying a filter in the frequency domain to reinforce the frequency less than 250 Hz. To improve the performance of the filter, I split the audio into six equal parts, each lasting for around 10 seconds. We can see that with this filter, the notes above Middle A (200 Hz) are almost eliminated. However, the filter is not perfect since the drum would also produce similar frequencies. It might be proved to apply multiple filters to get rid of the drums.

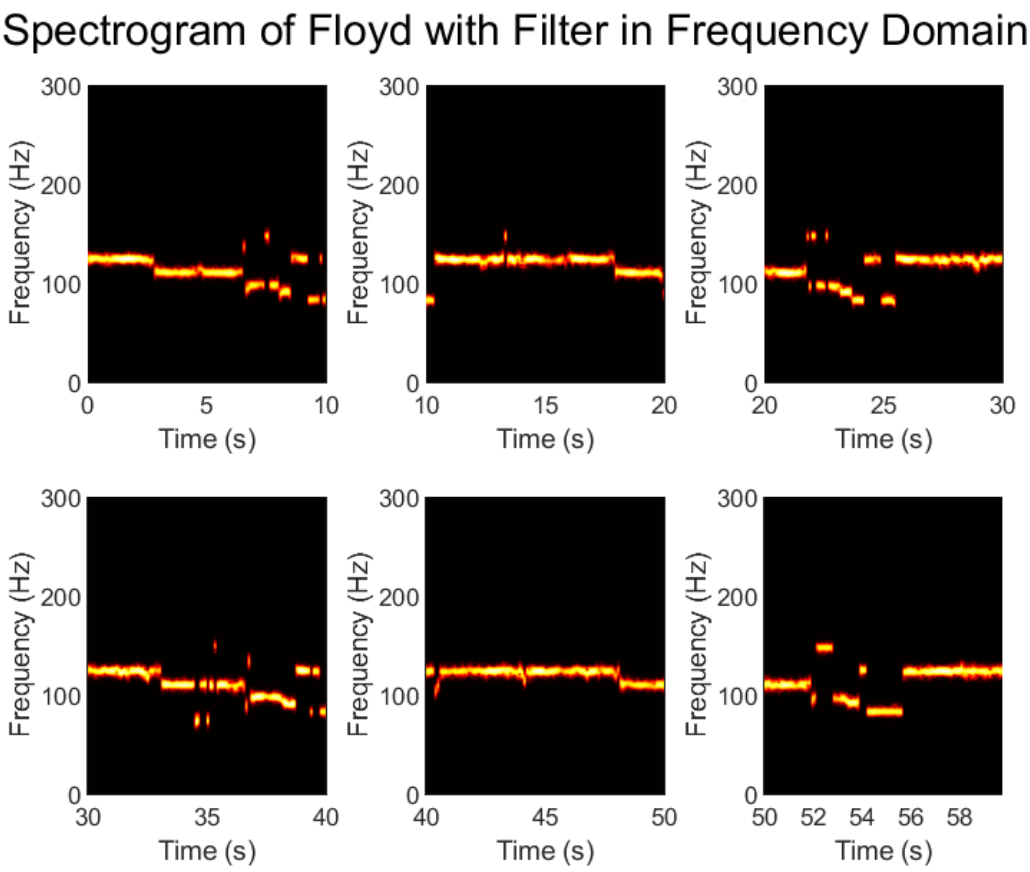

Figure 3. The spectrogram of the bass in the Floyd clip with a filter of 250 Hz

(c) Guitar Solo in the Floyd Clip

Figure 4. is the spectrogram of the guitar solo in the Floyd clip after applying two filters in the frequency domain: the first one is to undermine the frequency less than 250 Hz, which is the maximum frequency produced by the bass and the drum, and the second one is to reinforce the frequency less than 1200 Hz, which is the highest frequency produced by the guitar. Similar to part (b), I split the audio into six equal parts, each lasting for around 10 seconds, to improve the quality of the filter. The bright lines between 300 Hz and 800 Hz seem satisfactory, but we still haven't eliminate overtones. As suggested in the algorithm implementation, if we could build a filter by summing the center frequency of overtones up, the plot might be more concentrated. Another problem is that there are still bright dots or lines below 250 Hz, changing the parameter in the filter or using other filters might be a method to solve this issue.

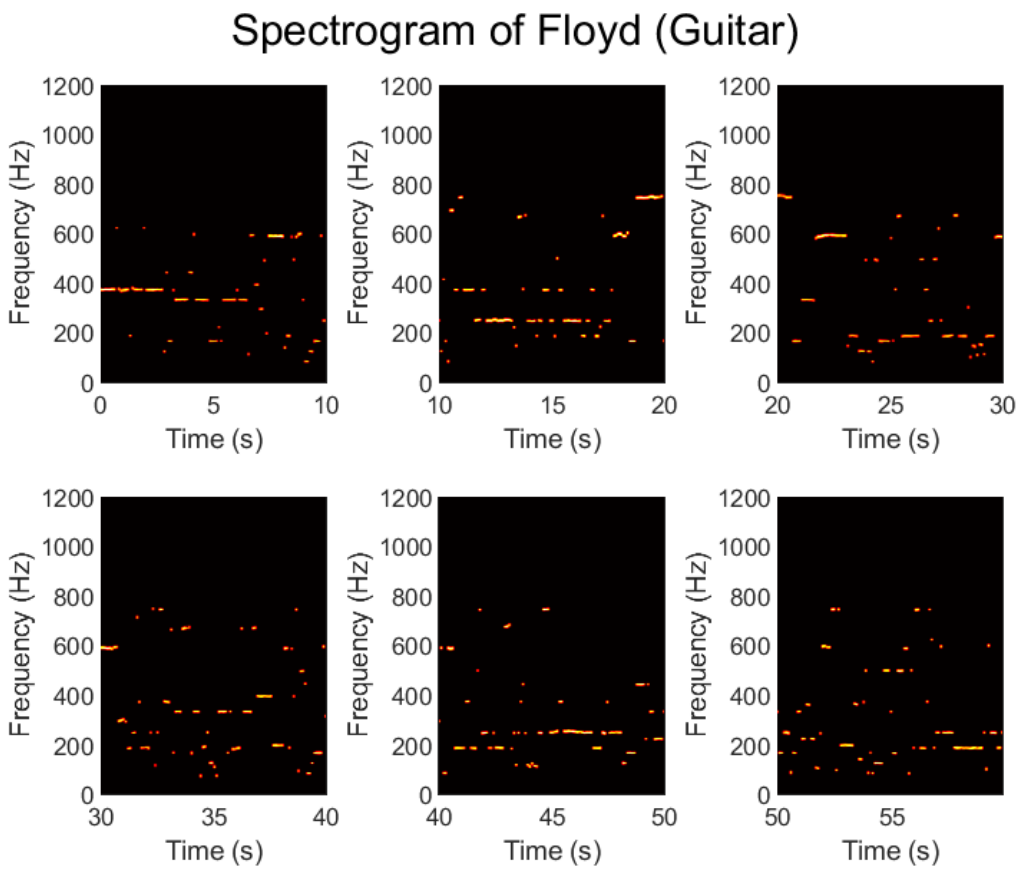

Figure 4. The spectrogram of the guitar in the Floyd clip

V. Summary and Conclusions

In this assignment, we mainly isolated the guitar and the bass from the GNR clip and the Floyd clip through Gabor transform and applying a Gaussian filter in the frequency domain. We also found out that applying a filter under certain frequency is useful and effective in isolating the instrument. However, there are some issues left as well. Overtones and similar frequency are extremely difficult to filter out. The solutions might be applying multiple filters, changing the parameter of the filter, or implement other filters.

Appendix A. MATLAB functions[1]

`[y, Fs] = audioread('filename')`: reads data from the file named `filename`, and returns sampled data, `y`, and a sample rate for that data, `Fs`. We used it to load our audio.

`length(X)`: returns the length of the largest array dimension in `X`. We used it to find the whole time interval of the audio.

`linspace(x1,x2,n)`: Generates `n` points with an even spacing between `x1` and `x2`. We used it to generate an evenly spaced time interval.

`fftshift(X)`: Rearranges a Fourier transform `X` by shifting the zero-frequency component to the center of the array. We used it to fix the visualization of the transformed data.

`zeros(n1,n2)`: Returns an `n1`-by-`n2` array of zeros. We used it to create new arrays to store outputs.

`abs(X)`: Returns the absolute value of each element in array `X`. Since `X` is complex here, we used it to return the complex magnitude.

`fftn(X)`: Returns the multidimensional Fourier transform of an N-D array `X` using a fast Fourier transform algorithm. We used it to transform our data into the frequency domain.

`[M,I] = max(A)`: Returns the maximum `M` and the corresponding index `I` of matrix `A`. We used it to gain the index in order to find the center frequency using `ind2sub`.

`[x,y] = ind2sub(sz,ind)`: Returns the coordinates (`x,y`) given an index `ind` of a matrix of size `sz`. We used it to find the coordinates of the center frequency.

`pcolor(X,Y,C)` specifies the `x`- and `y`-coordinates for the vertices. The size of `C` must match the size of the `x-y` coordinate grid. Here, we used it to plot the log-frequency spectrogram at given coordinates.

[1] Basic function descriptions from MathWorks Help Center.

Appendix B. MATLAB codes

```
%% given code
% figure(1)
% [y, Fs] = audioread('GNR.m4a');
% tr_gnr = length(y)/Fs; % record time in seconds
% plot((1:length(y))/Fs,y);
% xlabel('Time [sec]'); ylabel('Amplitude');
% title('Sweet Child O Mine');
% p8 = audioplayer(y,Fs); playblocking(p8);

%% music score for GNR
clear all; close all; clc;
[y, Fs] = audioread('GNR.m4a');
tr_gnr = length(y)/Fs; % record time in seconds
L = tr_gnr;
n = length(y);
t1 = linspace(0,L,n+1);
t = t1(1:n);
k = (2*pi/L)*[0:n/2-1, -n/2:-1];
ks = fftshift(k);

% Gabor filter
step = 100;
t_g = linspace(0, t(end), step);
spec = zeros(length(t_g), n);

for i=1:length(t_g)
    gabor = exp(-100 * (t - t_g(i)).^2);
    gt = fft(gabor .* y');
    gts = abs(fftshift(gt));
    [gts_max, ind] = max(gts(n/2:end));
    [a,b] = ind2sub(size(gts),ind+n/2-1);
    filter = exp(-0.001 * (ks - ks(b)).^2);
    spec(i,:) = abs(fftshift(gt).* filter);
end

% spectrogram
figure(1)
pcolor(t_g, ks/(2*pi), log(spec'+1)), shading interp
colormap('hot'), xlabel('Time (s)'), ylabel('Frequency (Hz)')
axis([0,tr_gnr,0,1500])
title('Spectrogram of GNR')
```

```
%% music score for Floyd
clear all; close all; clc;
[y, Fs] = audioread('Floyd.m4a');
y = y(1:end-1, 1); % reject the last data to avoid error
tr_floyd = length(y)/Fs; % record time in seconds
L = tr_floyd;
n = length(y);
t1 = linspace(0,L,n+1);
t = t1(1:n);
k = (2*pi/L)*[0:n/2-1, -n/2:-1];
ks = fftshift(k);

% Gabor filter
step = 100;
t_g = linspace(0, t(end), step);
spec = zeros(length(t_g), n);

for i=1:length(t_g)
    gabor = exp(-100 * (t - t_g(i)).^2);
    gt = fft(gabor .* y');
    gts = abs(fftshift(gt));
    [gts_max, ind] = max(gts(n/2:end));
    [a,b] = ind2sub(size(gts),ind+n/2-1);
    filter = exp(-0.001 * (ks - ks(b)).^2);
    spec(i,:) = abs(fftshift(gt).* filter);
end

% spectrogram
figure(2)
pcolor(t_g, ks/(2*pi), log(spec'+1)), shading interp
colormap('hot'), xlabel('Time (s)'), ylabel('Frequency (Hz)')
axis([0,tr_floyd,0,500])
title('Spectrogram of Floyd')

%% filter in frequency domain = 250Hz
clear all; close all; clc;
[y, Fs] = audioread('Floyd.m4a');
itv = 10; % interval = 10 sec per plot

figure(3)
for i=1:length(y)/(Fs*itv)+1
    if i*itv*Fs < length(y) % load part
        [y_s, Fs] = audioread('Floyd.m4a', [(i-1)*itv*Fs+1, i*itv*Fs]);
    else
```

```
        [y_s, Fs] = audioread('Floyd.m4a',
                              [(i-1)*itv*Fs+1, length(y)-1]);
    end

    tr_floyd = length(y_s)/Fs; % record time in seconds
    L = tr_floyd;
    n = length(y_s);
    t1 = linspace(0, L, n + 1);
    t = t1(1:n);
    k = (2*pi/L)*[0:n/2-1, -n/2:-1];
    ks = fftshift(k);

    % Gabor filter
    step = 100;
    t_g = linspace(0, t(end), step);
    spec = zeros(length(t_g), n);

    for j=1:length(t_g)
        gabor = exp(-100 * (t - t_g(j)).^2);
        gt = fft(gabor.*y_s');
        gts = abs(fftshift(gt));
        [gts_max, ind] = max(gts(n/2:n/2+2*pi*250)); % frequency = 250
        [a,b] = ind2sub(size(gts),ind+n/2-1);
        filter = exp(-0.001 * (ks - ks(b)).^2);
        spec(j,:) = abs(fftshift(gt).* filter);
    end

    % spectrogram
    subplot(2,3,i)
    pcolor(t_g+itv*(i-1), ks/(2*pi), log(spec'+1)), shading interp
    colormap('hot'), xlabel('Time (s)'), ylabel('Frequency (Hz)')
    axis([itv*(i-1), tr_floyd+itv*(i-1), 0, 300])
    drawnow
end
sgtitle('Spectrogram of Floyd with Filter in Frequency Domain')

%% guitar solo
clear all; close all; clc;
[y, Fs] = audioread('Floyd.m4a');
itv = 10; % interval = 10 sec per plot

figure(4)
for i=1:length(y)/(Fs*itv)+1
    if i*itv*Fs < length(y) % load part
```

```
        [y_s, Fs] = audioread('Floyd.m4a', [(i-1)*itv*Fs+1, i*itv*Fs]);
    else
        [y_s, Fs] = audioread('Floyd.m4a',
                              [(i-1)*itv*Fs+1, length(y)-1]);
    end

    tr_floyd = length(y_s)/Fs; % record time in seconds
    L = tr_floyd;
    n = length(y_s);
    t1 = linspace(0, L, n + 1);
    t = t1(1:n);
    k = (2*pi/L)*[0:n/2-1, -n/2:-1];
    ks = fftshift(k);

    % Gabor filter
    step = 100;
    t_g = linspace(0, t(end), step);
    spec = zeros(length(t_g), n);

    % filter out bass & drum
    for j=1:length(t_g)
        gabor = exp(-100 * (t - t_g(j)).^2);
        gt = fft(gabor .* y_s');
        gts = abs(fftshift(gt));
        [gts_max, ind] = max(gts(n/2:n/2+2*pi*250)); % freq of bass/drum
        [a,b] = ind2sub(size(gts),ind+n/2-1);
        filter = exp(-0.001 * (ks - ks(b)).^2);
        gts = abs(fftshift(gt).*(1 - filter));

        % filter for the guitar
        [gts_max, ind] = max(gts(n/2:n/2+2*pi*1200)); % freq of guitar
        [a,b] = ind2sub(size(gts), ind+n/2-1);
        filter = exp(-0.001 * (ks - ks(b)).^2);
        spec(j,:) = abs(fftshift(gt).* filter);
    end

    % spectrogram
    subplot(2,3,i)
    pcolor(t_g+itv*(i-1), ks/(2*pi), log(spec'+1)), shading interp
    colormap('hot'), xlabel('Time (s)'), ylabel('Frequency (Hz)')
    axis([itv*(i-1), tr_floyd+itv*(i-1), 0, 1200])
    drawnow
end
sgtitle('Spectrogram of Floyd (Guitar)')
```