AMATH 582 Homework 2

Joseph David

February 10, 2020

Abstract

We write a `MATLAB` program to analyze two Rock 'n Roll riffs, `GNR.m4a` (Guns N Roses' *Sweet Child O' Mine*) and `Floyd.m4a` (Pink Floyd's *Comfortably Numb*). In particular, we use the Gabor Transform to determine the dominant frequency of the riffs over time, which will allow us to determine which notes are being played and produce a "score" for each one.

1 Introduction and Overview

The two files `GNR.m4a` and `Floyd.m4a` are audio files that each contain the signal of a segment of their respective songs. Thus, we convert each audio file into a vector that contains the amplitude of the signal sampled at a regular interval (see Figure 1). Therefore, we can utilize the Fourier Transform in order to determine the dominant frequencies in the riffs. However, since the notes being played change over time (otherwise it wouldn't quite be a song!) the frequencies will change over time. Thus, we use the Gabor Transform, where we filter the signal at a regular interval and, for each time, compute the Fourier Transform (see Figures 2 and 3). The collection of these Fourier Transforms are then used to create a *spectrogram*, which is a graph that shows the dominant frequencies at each point that we applied the filter. From this, we determine the sequence of notes being played (Figures 4 and 5).

2 Theoretical Background

The primary theory we use is the Fourier Transform (FT). Recall that the FT, $\widehat{f}(k)$, of a function $f(x)$ is defined by

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

This gives us information on which frequencies are present in the song, but does not tell us anything about *when* those frequencies occur. Therefore we use the Gabor Transform (GT), where we filter our signal at various points in time in order to obtain information on when the frequencies occur. Letting $g(\tau)$ denote our filter, the definition of the GT of a function $f(x)$, is given by

$$\widehat{f}_g(t,\omega) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)e^{-i\omega\tau}d\tau. \tag{2}$$

Notice that (1) and (2) are very similar. Multiplying by the filter $g(t-\tau)$ essentially isolates our signal near t, so for various t's in our time domain, we obtain a Fourier Transform of the only the part of the signal that occurs near that time. Thus, we obtain frequency information on our signal at different points in time. This will allow us to determine the sequence of notes in our music.

One issue with the Gabor Transform is that we have two degrees of freedom, how finely to slide our filter along the time domain and how wide to make our filter. We must make decisions here that will make sense for what we are trying to accomplish, which is isolate the notes.

If our filter is too narrow, the lower frequencies might not be picked up and if it is too large then it may span too much time which would make it difficult to isolate notes. On the other hand, if we do not slide our filter at enough points then we may miss notes, but we do not want to slide so finely that our code takes

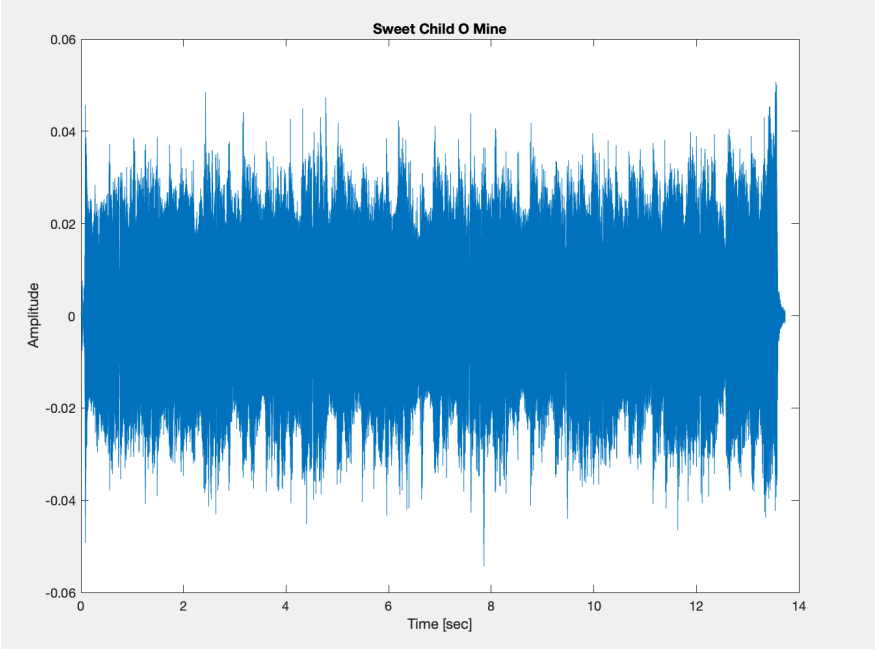

Figure 1: Signal for the beginning of *Sweet Child O' Mine* by Guns N Roses, contained in the audio file `GNR.m4a`.

very long to run. Thus, in order to choose these parameters well, it is helpful to know something about the frequencies of the notes and how long they last.

In GNR we are looking for relatively high frequency guitar notes [3], of which each is only played for a short amount of time (about four per second). Thus, we choose a narrow filter and slide the filter by about an eigth of a second each time, to ensure each note is captures. In `Floyd`, we are looking for bass notes with a lower frequency [1] and fewer notes so we may take a slightly wider filter to eliminate the guitar notes and we can slide our filter by increments of only one second.

Lastly, in order to read the notes from our spectrograms, we refer to [2] to determine the notes given the frequency.

3 Algorithm Implementation and Development

Algorithm 1: Creating a Spectrogram

Import song data from `GNR.m4a/Floyd.m4a`
Create a filter with desired width and time vector to slide filter
Create matrix `yftspec` that will contain the frequency information over time
for $j = 1 : length(timevector)$ **do**
  Center filter at time vector(j)
  Multiply song signal by filter
  Take the FT (and for `Floyd` also multiply by bass filter to isolate lower frequencies)
  Store it in new row of `yftspec`
end for
Plot spectrogram of `yftspec`

Implementation of Algorithm 1 We use a Gaussian `exp(-a(t-tslide))` for our filter and `tslide=x:b:y` to define the time vector, a vector of the numbers between x and y in increments of b. Thus the two variables `a` and `b` are used to control the width of our filter and time increment in seconds, respectively. For `GNR.m4a`,

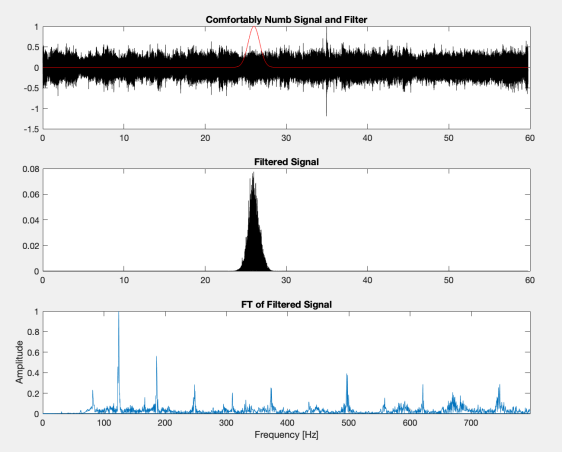

Figure 2: Applying filter around 25 seconds

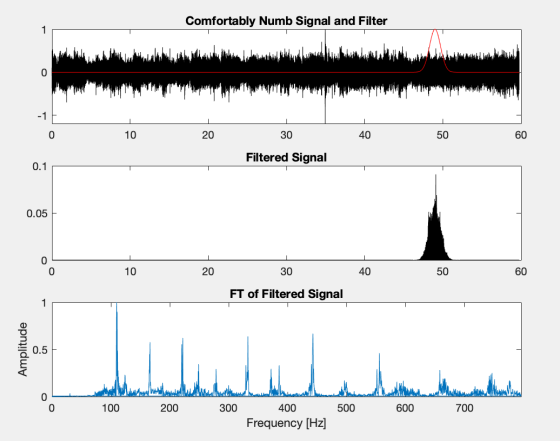

Figure 3: Filter moves to around 50 seconds

we use `a=30` and `b=.125` in order to isolate the fast notes being played. For `Floyd` bass, we use `a=1` and `b=1` since we need to capture low frequencies and the notes are played for a length of time on the order of a second, while we use `a=100` and `b=.3` for `Floyd` guitar. For the bass filter, we define a Guassian in the frequency (in Hz) vs time domain, centered at 150Hz and with width parameter .001, which essentially captures frequencies between 80 and 220 Hz, and we use another Gaussian for the guitar filter centered at 450Hz and with width parameter .00002, which captures frequencies roughly from 150-750Hz. We use `fftn` to compute the Fourier Transform of the filtered signal at each time and `pcolor` to plot the spectrogram.

4 Computational Results

See Figures 4, 5, and 6 for the spectrograms of Sweet Child O Mine, Comfortably Numb bass and Comfortably Numb guitar, respectively. The corresponding notes are marked. We referred to [2] to determine from the given frequency the corresponding musical note.

As can be seen, the spectrograms in Figures 4 and 5 are quite clear, while in 6 it is not. This is perhaps because the sound is much more muddled in the middle frequency region. Therefore, the notes for this spectrogram are not labelled because it is not clear which frequencies represent the guitar and which are overtones or other instruments.

We discuss the different modifications that were tried in order to produce the best spectrogram for the guitar in `Floyd`. In order to make the guitar stand out, we used a filter in frequency space with the purpose of lowering the frequency values of the bass, which is particularly strong in `Floyd`. This saw the best performance, but still was not ideal.

We therefore tried other methods, such as making the filter width in the Gabor Transform extremely small, with the aim of not allowing the low bass frequency information to pass through. However this had the negative impact of "smoothing" out the Fourier Transform and thus the frequency information was not specific enough. We also tried "compressing" the audio file, as in taking all signal above a certain amplitude and scaling it down. The idea was that perhaps the biggest amplitudes were caused by the bass and by lowering these parts, we would see smaller values for the bass frequencies in the Fourier Transform.

5 Summary and Conclusions

We were able to successfully create spectrograms of the audio files `GNR.m4a` and `Floyd.m4a` using the Gabor Transform. The spectrograms for the guitar in `GNR` (see Figure 4) and bass in `Floyd` (see Figure 5) were particularly clear, however the spectrogram for the guitar in `Floyd` (see Figure 6) was much less clear.

The biggest difficulty of this process was determining a good combination filter and time increment. For instance, in figure 5, the order of the notes is pretty clear, whereas in Figure 4 it is sometimes difficult to

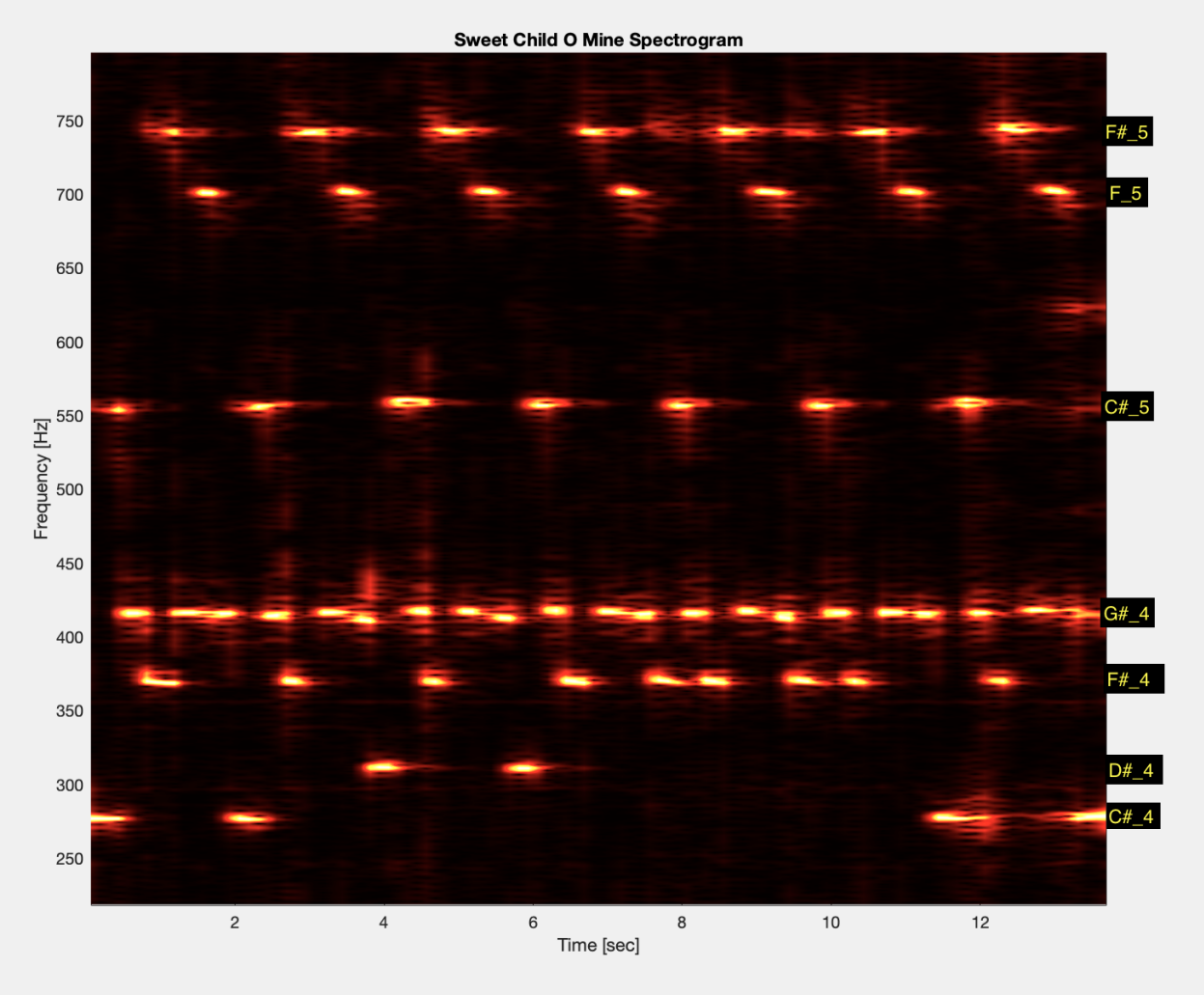

Figure 4: Spectrogram for *Sweet Child O' Mine* by Guns N Roses. Shows frequencies of about 200-800Hz and the corresponding musical notes.

see exactly the order of the notes. Part of this we cannot fix because sometimes there are two notes being heard at the same time. However, if we had used a narrower filter and finer time increment we may have been able to produce a better spectrogram for `GNR`, but in the interest of computational expense the one given was good enough.

References

[1] *Bass Frequency Range*. URL: `https://www.studybass.com/gear/bass-tone-and-eq/bass-frequency-range/`.

[2] *Frequencies of Musical Notes*. URL: `https://pages.mtu.edu/~suits/notefreqs.html`.

[3] Kurt Prange. *Audible Frequency Range and Describing Tone*. URL: `https://www.amplifiedparts.com/tech-articles/audible-frequency-range-and-describing-tone`.

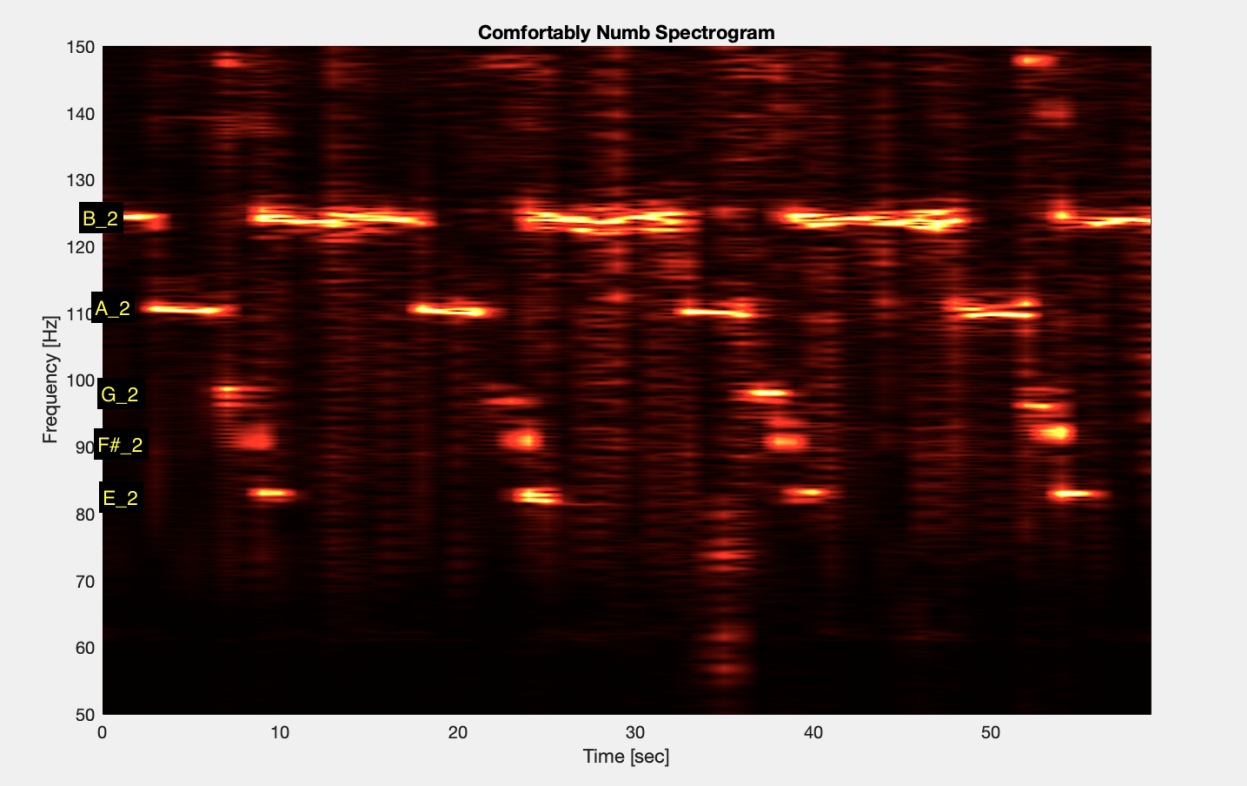

Figure 5: Spectrogram for *Comfortably Numb*. Only shows lower frequencies, up to 150Hz and the corresponding musical notes.

Appendix A MATLAB Functions

Add your important MATLAB functions here with a brief implementation explanation. This is how to make an **unordered** list:

- `audioread('Song.m4a')` returns a row vector with the song signal and the sampling rate.
- `x:b:y` returns a row vector with numbers between `x` and `y` in increments of length `b`.
- `g = fftn(f)` returns the mulitdimensional Fourier Transform of f, where f is a multidimensional array.
- `y = fftshift(x)` centers the matrix `x` around the zero frequency. It is commonly used after computing the Fourier Transform, so that your data is centered around the zero frequency.
- `pcolor(A)` returns an grid image of the matrix `A` where each entry is colored based on it value.

Appendix B MATLAB Code

See the two `MATLAB` programs below.

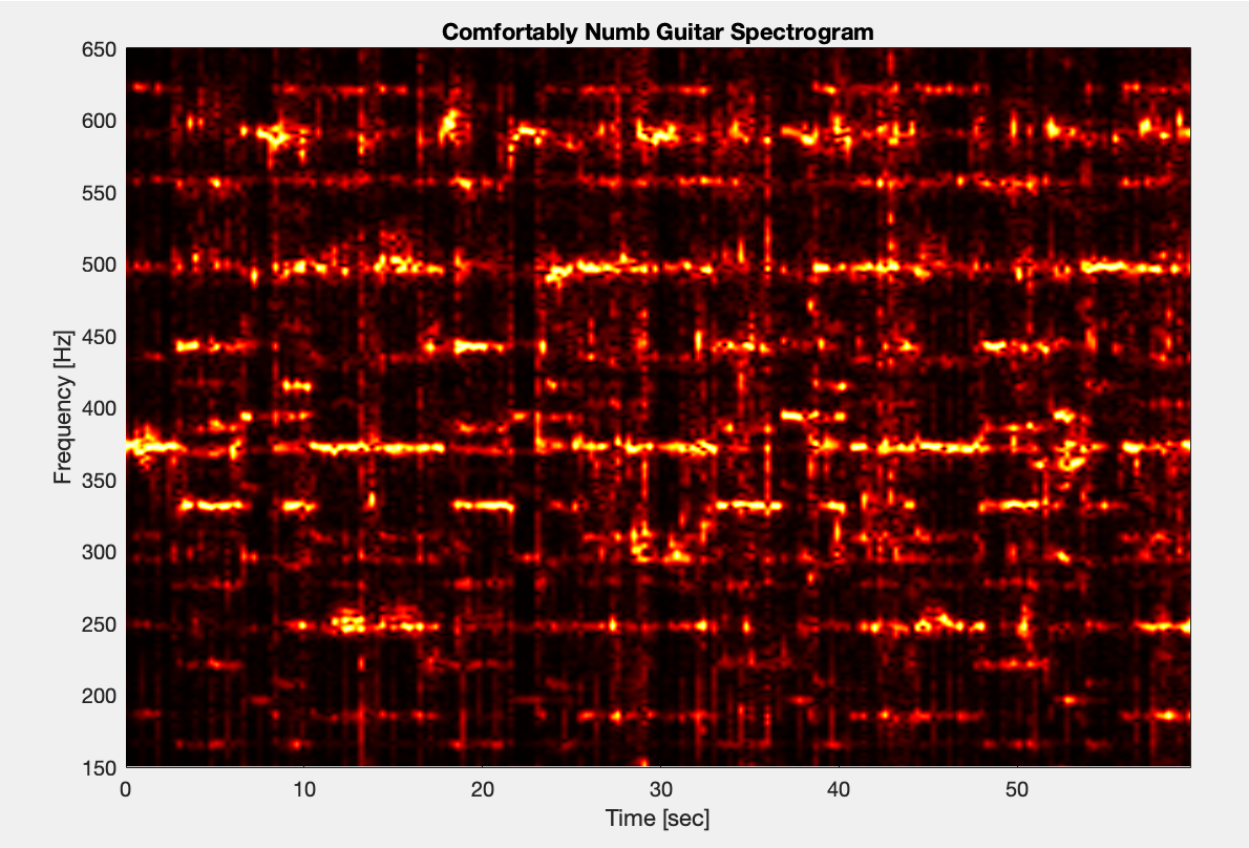

Figure 6: Spectrogram for the guitar solo in *Comfortably Numb*.

```
clear all; close all; clc

figure(1)
[y, Fs] = audioread('GNR.m4a');
trgnr = 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);

n = length(y);
t = (1:n)/Fs;

k = (2*pi/trgnr) * [0:(n/2)-1 -n/2:-1]; ks = fftshift(k);
% fy = fftshift(fft(y));
% figure(2)
% plot(ks,fy);

a = 30;  %fineness of filter
b = .125; %fineness of slide
yft_spec = [];
ksfreq = ks*(1/(2*pi));
tslide = b/2 : b : t(n);

freq = zeros(1,length(tslide));

for j = 1 : length(tslide)
    filter = exp(-a*((t-tslide(j)).^2));
    yf = y.' .* filter;
    yft = fft(yf);

    yft_spec = [yft_spec; abs(fftshift(yft))/max(abs(yft))];

    figure(2)
    subplot(3,1,1), plot(t, y, 'k', t, filter, 'r')
    subplot(3,1,2), plot(t, abs(yf), 'k')
    subplot(3,1,3), plot(ks, abs(fftshift(yft))/max(abs(yft)))
    axis([-5000 5000 0 1])

end

figure(3)
pcolor(tslide, ksfreq(n/2+3000:n), yft_spec(:,n/2+3000:n).'), shading interp
set(gca, 'Ylim',[218 5000/(2*pi)])
title('Sweet Child O Mine Spectrogram')
xlabel('Time [sec]'); ylabel('Frequency [Hz]')
colormap(hot)
```

Listing 1: Code for the Guns N Roses portion.

```
clear all; close all; clc

figure(1)
[y, Fs] = audioread('Floyd.m4a');
trgnr = length(y)/Fs; % record time in seconds
n = length(y);
t = (1:n)/Fs;

k = (2*pi/trgnr) * [0:(n/2)-1 -n/2:-1]; k(n)=0; ks = fftshift(k);

a = 1;  %fineness of filter
b = 1; %fineness of slide

a2 = .001; %bass filter width
b2 = 150; %bass filter center
bfilter = exp(-a2*((ksfreq-b2).^2)); %bass filter
yft_spec = [];
ksfreq = ks/(2*pi); %gives frequencies
tslide = 0 : b : t(n);
freq = zeros(1,length(tslide));

for j = 1 : length(tslide)
    filter = exp(-a*((t-tslide(j)).^2));
    yf = y.' .* filter;
    yft = bfilter .* fftshift(fft(yf)); %filter out higher frequencies
    yft_spec = [yft_spec; abs(yft)/max(abs(yft))];
end

figure(3)
pcolor(tslide, ksfreq((n-1)/2 : 1327500), yft_spec(:, (n-1)/2 : 1327500).'), shading interp
set(gca, 'Ylim',[50 150])
title('Comfortably Numb Bass Spectrogram')
xlabel('Time [sec]'); ylabel('Frequency [Hz]');
colormap(hot)

%determine the guitar part
a3 = .00002; b3 = 450;
gfilter = exp(-a3*((ksfreq-b3).^2));
yft_spec = [];
a=100;
b = .3;
tslide = 0 : b : t(n);
for j = 1 : length(tslide)
    filter = exp(-a*((t-tslide(j)).^2));
    yf = y.' .* filter;
    yt = fftshift(fft(yf));
    yft = yt .* gfilter; %filter out lower frequencies
    yft_spec = [yft_spec; abs(yft)/max(abs(yft))];
end

figure(4)
pcolor(tslide, ksfreq((n-1)/2+9000:(n-1)/2+45000), yft_spec(:,(n-1)/2+9000:(n-1)/2+45000).'), shading in
set(gca, 'Ylim',[150 650])
title('Comfortably Numb Guitar Spectrogram')
xlabel('Time [sec]'); ylabel('Frequency [Hz]');
colormap(hot)
```


Listing 2: Code from the Pink Floyd portion.