Time-Frequency Analysis: Identifying Music Notes

Christine M. Baker, GitHub: `cmbaker94`

Feb 10, 2021

Abstract

Gábor transforms, a powerful tool for resolving time and frequency information from data, was used to identify notes in two famous rock songs. Spectrograms with the frequency axis relabeled for the related musical notes for the guitar in both songs and bass in one song. When only a guitar solo was present, the spectrogram clearly had peaks at frequencies related to the guitar musical notes. Whereas, when several instruments were included, it was difficult to distinguish the guitar from the drums and overtones of the bass.

1 Introduction and Overview

The music scale from clips of the songs *Sweet Child O' Mine* by Guns N' Roses and *Comfortably Numb* by Pink Floyd are identified using Gabor filtering. By varying the Gabor window size and number of windows, a spectogram of the clips is produced with peaks at frequencies related to the guitar and bass notes played in the song. Boxcar low-pass and bandpass filters are applied to the spectra to retain only the frequencies of interest and variance thresholds are used to produce clear peaks in the spectogram plots.

The theoretical background on windowed fourier transforms, specifically Gabor windows is provided in Section 2. The alogrithm implementation and development to identify the notes in both songs is presented in Section 3. The computational results including guitar and bass notes are shown in Section 4. A conclusions are summarized in Section 5. Details about the MATLAB functions and the function writen for this project are in Appendix A and the MATLAB code to run the analysis is in Appendix B.

2 Theoretical Background

Although Fourier transforms are useful for characterizing a stationary or periodic signal, Fourier transforms eliminate all time-domain information and are therefore limited in capturing the moment in time when variance at specific frequencies occur (Kutz, 2013). For non-stationary signals with a constant time-average signal value, time-frequency information can be attained by decomposing the signal into smaller windows.

Gábor Dénes, a Hungarian-British electrical engineer and physicist, purposed a method to localize time and frequency by modifying the Fourier transform kernel. With a Gábor kernel

$$g_{t,\omega}(\tau) = e^{i\omega\tau} g(\tau - t)$$

and with the assumption that g is real and symmetric with $||g(t)|| = 1$ and $||g(\tau - t)|| = 1$, the *Gábor transform* can be modified as:

$$G[f](t,\omega) = \tilde{f}_g(t,\omega) = \int_{-\infty}^{\infty} f(\tau)g(\tau - t)e^{-i\omega\tau}d\tau$$

with the term $g(\tau - t)$ acts as a filter in time, localizing data over a time window centered at $t = \tau$. The width of the window, a can be modified to include longer or shorter time windows.

This method reduces some accuracy in time and frequency to simultaneously obtain both time and frequency resolution. A major drawback of this approach is the loss of low frequency variance, which cannot be resolved since the wavelength of this signal is longer than the window.

3 Algorithm Implementation and Development

The Gábor Transform is applied to an amplitude signal of *Sweet Child O' Mine* by Guns N' Roses and *Comfortably Numb* by Pink Floyd. To identify when musical notes were played during a song, time-domain information is required. Therefore, Gábor transform was used to resolve the frequency and time domain information for each song. Since the guitar and/or bass have wavelengths much smaller than the time series length, windowing the time series did not remove any important information.

Since the Gábor transform was applied to two songs and for different frequencies of interest, a function was created and is included in Appendix A. The code computes a spectrogram with Gábor transforms given a time series, frequency of the time series, time step of the Gábor transform, and width of the window. Additionally, the boxcar bandpass filter range for the output spectra is included as a 2x1 vector with the low and high ends of the filter. Lastly, an indicator telling if a figure should be plotted for each transform is included. The time step changed the time-resolution of the spectrogram and the window width changed the lowest frequency resolved. The function computes the length of the record and generates a time and frequency vector. Unlike the spatial data, the wavenumber does not have a 2π factor since Hz is 1/s. The time vector is computed as a vector from zero to the seconds of data with the defined time step. The indexes of the frequency bounds for the bandpass filter of each spectra are established and applied to the frequency vector that is output from the function.

The Gábor Transform is then computed for the data based on the time resolution and width, a. The filter is computed as $g = \exp(-a(t - \tau)^2)$, where τ is the time the filter is centered at (*e.g.,* Figure 1a). Then, the filter is multiplied by the data (*e.g.,* Figure 1b) and a Fast Fourier Transform is computed for the Gábor windowed data and the spectra is shifted with the fftshift function (*e.g.,* Figure 1c). Lastly, the bandpass filter is applied to the spectra and the data is stored. For the purposes of this analysis, instead of storing all the data for the frequencies set to zero for the bandpass filter, the data is trimmed to decrease the size of the spectrogram matrix. Once all the data is looped through, the function outputs the spectrogram, frequencies, and time where each spectra is centered at.

This function is then applied to the *.m4a data for both songs, which is read in with an audioread function. The entire time series is used for the Guns N' Roses (GNR) song, whereas the Pink Floyd (PF) amplitude vector was trimmed by one point for the purposes of computing the Fourier Transform. The normalized spectrogram is thresholded to emphasize the variance as the musical notes. The width, time resolution, bandpass filter bounds, and threshold value varied per song and instrument of interest are presented in Table 1. Bandpass filter bounds were chosen based on conservative guitar and bass frequency range of 250-800 Hz and 0 - 250 Hz, respectively, where filtering to 0 Hz is analogous to a low-pass filter. The threshold value was selected based on visual identification of the best produced spectrogram of the musical notes. Note that the filter width must be long enough (input value is small enough) to resolve the lower frequencies of the bass. The complete code for this analysis is shown in Appendix B.

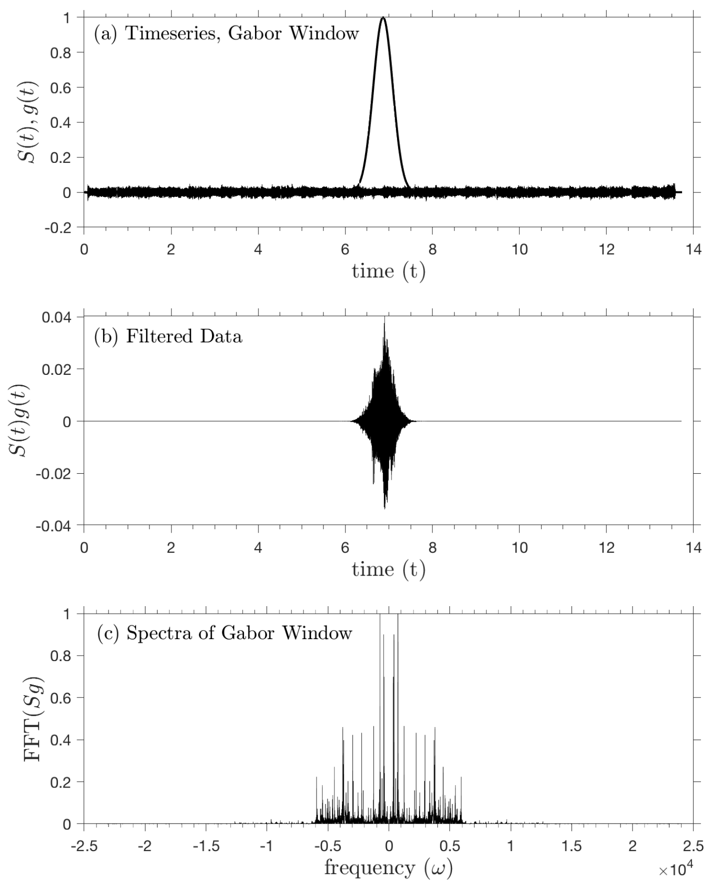

Figure 1: An example of the Gábor window applied to a data time series (a). When the data is multiplied by the filter, only a small portion of windowed data remains (b) and used to compute a spectra (c).

4 Computational Results

The guitar solo in *Sweet Child O' Mine* by Guns N' Roses (GNR) was easily distinguishable due to little background variance in the time series and the threshold cleanly removed the background variance from the time series to produce a clean spectrogram (Figure 2). Notes in the song included C_4 sharp (261.63 Hz), F_4 sharp, G_4 sharp, C_5 sharp, F_5, and F_5 sharp (739.99 Hz). The range of notes were bandpassed between 250 - 800 Hz.

Table 1: The song, instrument of interest, filter width, spectrogram resolution defined by the Gábor transform time step, bandpass low and high filter founds, and the normalized threshold used.

Song	Instrument	width	resolution (sec)	filter bounds (Hz)	threshold
GNR	guitar	500	0.1	[250,800]	0.25
PF	bass	100	0.5	[0, 250]	0.24
PF	guitar	50	0.5	[250, 800]	0.2

The 60 sec clip of *Comfortably Numb* by Pink Floyd (PF) has several instruments including bass, guitar, and drums. This made individual instruments more difficult to decipher between due to more variance including overtones at many frequencies. The bass, between 50 - 250 Hz, was more easily distinguishable by finding an appropriate Gábor window width (Figure 3b) and included the notes: E_2, G_2, A_2, and B_2. The guitar was more difficult to identify specific notes due to overtones from the bass. Originally, a filter range similar to GNR was used for the guitar; however, it appeared that the overtones of the bass showed up as a multiple of 3 of the frequencies of the bass between 250-370 Hz. After listening to the music several times, it sounded like the guitar was also producing sounds at the same beat as the bass. Therefore, the lower frequency range was included for the PF guitar range, due to my inability to audibly distinguish the exact notes played by the guitar. A D_5 note was clearly repeated in the guitar section of PF among other notes.

5 Summary and Conclusions

Musical notes in classic rock songs can be identified using the Gábor transform for *Sweet Child O' Mine* by Guns N' Roses (GNR) and *Comfortably Numb* by Pink Floyd (PF). The Gábor transform provided a tool to resolve both frequency and time information with a Fourier analysis. By filtering ranges based on the instrument and thresholding returns, notes could be shown using a spectrogram. The guitar solo in in GNR was easily distinguishable due to minimal background noise, whereas the guitar beats in PF were more difficult to readily remove because two other instruments were included in the song. The bass in PF had more easily identifiable notes due to the low frequencies which did not come from any of the other instruments.

Appendix

A MATLAB functions used and brief implementation explanation

- `abs`: returns the absolute value of the input element
- `audioread`: reads an m4a and produces a vector of amplitude and defines the frequency
- `audioplayer`: make an audoplayer variable

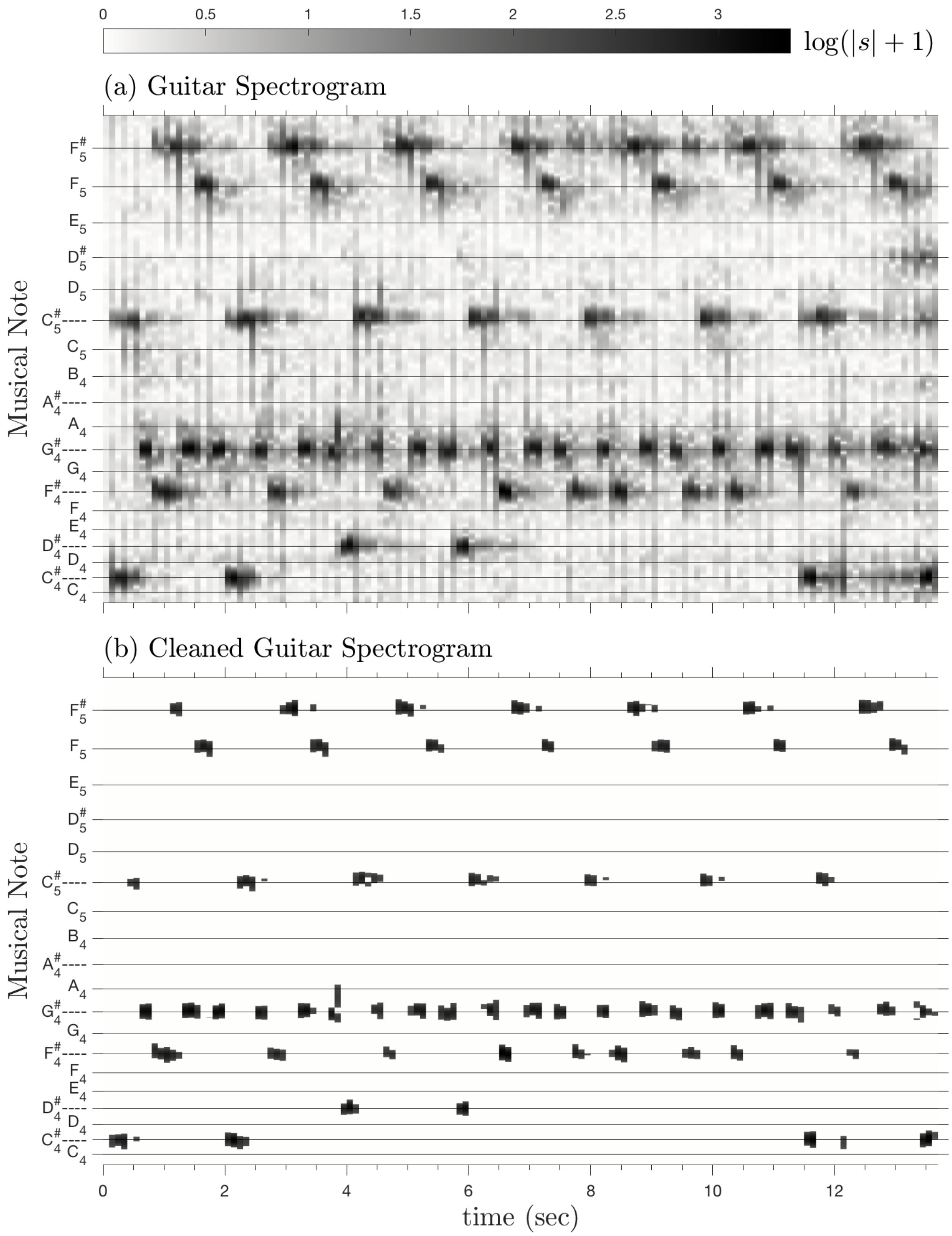

Figure 2: Guitar range spectrogram of the Gábor transformed variance from GNR with overtones removed via a bandpass boxcar filter (a) and cleaned with a threshold (b). The frequencies are replaced by the note names associated with those frequencies (y-axis) for the ∼13 sec clip of the song (x-axis).

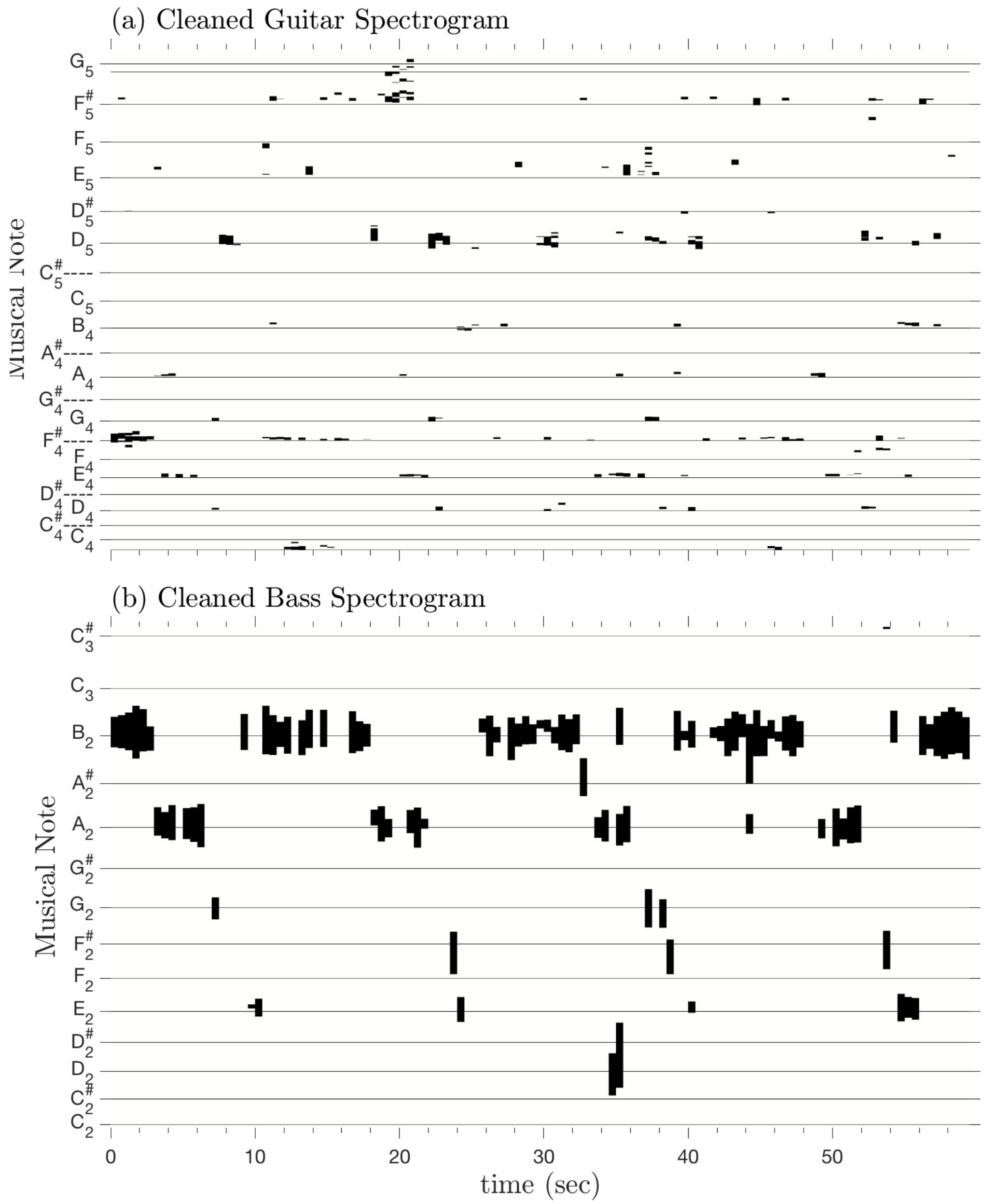

Figure 3: Spectrogram of the Gábor transformed variance from PF with a bandpass boxcar filter at the guitar (a) and bass (b) frequencies and a threshold applied to clean the spectrogram. The frequencies are replaced by the note names associated with those frequencies (y-axis) for the 60 sec clip of the song (x-axis).

- `axis`: indicate the limits for the current plotted axes
- `cmocean`: package of colormaps (used for the scatter plot)
- `exp`: the expoential e^x for each element in the array (used for the Gaussian filter)
- `fft`: 1D fast Fourier transform, returns a multidimensional Fourier transform of an array using a fast Fourier transform
- `fftshift`: used to rearrange zero-frequency component to the center of the spectrum
- `figure`: open new figure
- `gabor`$_{f}ilt : filterdata, seeexplanationinSection3andcodeattheendofAppendixA$`grid` $: addgridlinestoa$
- `*label`: label the x, y, z axes of plots
- `length`: find the length of the largest array dimension (grab the time-dimension in loops)
- `linspace`: generate linearly spaced vector (used to create vector of length dimensions)
- `load`: load data from a .m file into workspace (load subdata.m)
- `max`: find the maximum value of an array or matrix
- `min`: find the minimum value of an array or matrix
- `playblocking`: play audo variable
- `print`: export and save figure
- `set`: manipulate a figure
- `size`: grab size of a matrix
- `text`: define text for a figure

```
function [Sgt_spec,ks,tslide] = gabor_filt(S,Fs,dt,width,filtbound,
    pltfig)
% DESCRIPTION:
% This code will compute a spectrogram with a gabor window given a
    data
% timeseries and information about the gabor window. A boxcar filter
    is
% applied around the ranges of interest, where the spectrogram and
% wavenumbers are clipped around these regions to save memory.
% INPUT:
% S         = time series amplitude
% Fs        = frequency
% dt        = timestep of gabor transform
% width     = width of gabor window
% filtbound = low and high pass of boxcar filter
```

```
% pltfig    = indicator if plot should be created
% OUTPUT:
% Sgt       = spectogram
% ks        = frequencies
% tslide    = time gabor window centered on

L   = length(S)/Fs; % record time in seconds
n   = length(S);
t2  = linspace(0,L,n+1);
t   = t2(1:n);
k   = (1/L)*[0:n/2-1 -n/2:-1];
ks  = fftshift(k);
tslide = 0:dt:L;

% filter region
[val,id(1)] = min(abs(ks-filtbound(1)));
[val,id(2)] = min(abs(ks-filtbound(2)));
ks          = ks(id(1):id(2));

% prepare for transform
Sgt_spec=[];

if pltfig == 1
    figure
end

for j=1:length(tslide)
    g=exp(-width*(t-tslide(j)).^2); % Gabor
    Sg=g.*S;
    Sgt=fft(Sg);
    Sgtshift = abs(fftshift(Sgt));
    Sgt_spec=[Sgt_spec; Sgtshift(id(1):id(2))];
    if pltfig == 1
        subplot(3,1,1), plot(t,S,'k',t,g,'r')
        subplot(3,1,2), plot(t,Sg,'k')
        subplot(3,1,3), plot(ks,abs(fftshift(Sgt))/max(abs(Sgt)))
        drawnow
        pause(0.1)
    end
end
```

MATLAB codes

Code avaialble at: `https://github.com/cmbaker94/Baker`$_{A}MATH582$

```
% HW2 Analysis Code
```

```
clear all
close all
clc

addpath(genpath('/Users/cmbaker9/Documents/MTOOLS'))

%% STEP 0: Locate and load data

datapath    = '/Users/cmbaker9/Documents/UW_Classes/AMATH_Data_Analysis/HW2/';
figfolder   = '/Users/cmbaker9/Documents/UW_Classes/AMATH_Data_Analysis/HW2/figures/';
musicfiles  = {'GNR.m4a','Floyd.m4a'};
playsong    = 0; % 1 if play song, 0 if not play song

note.glet  = {'C_4','C_4^#----','D_4','D_4^#----','E_4','F_4','F_4^#----','G_4','G_4^#----',...
    'A_4','A_4^#----','B_4','C_5','C_5^#----','D_5','D_5^#','E_5','F_5','F_5^#','G_5','G_5^#'};
note.gHz = [261.63, 277.18, 293.66, 311.13, 329.63, 349.23, 369.99, 392.00, 415.30, 440.00, 466.16, 493.88, 523.25, 554.37, 587.33, 622.25, 659.25, 698.46, 739.99, 783.99, 830.61];

note.blet  = {'C_2','C_2^#','D_2','D_2^#','E_2','F_2','F_2^#','G_2','G_2^#',...
    'A_2','A_2^#','B_2','C_3','C_3^#','D_3','D_3^#','E_3','F_3','F_3^#','G_3',...
    'G_3^#','A_3','A_3^#','B_3'};
note.bHz = [65.41, 69.3, 73.42, 77.78, 82.41, 87.31, 92.5, 98, 103.83, 110, 116.54, ...
    123.74, 130.81, 138.59, 146.83, 155.56, 164.81, 174.61, 185, 196, 207.65, 220, 233.08, 246.94];


%% STEP 1: Read and filter Guns N' Roses to find guitar notes

[y, Fs]     = audioread([datapath,musicfiles{1}]);
S           = y';
playsong    = 0;

if playsong == 1
    p8 = audioplayer(S,Fs);
        playblocking(p8);
end

```

```
width    = 500;
dt       = 0.1;
pltfig   = 0;
filtbound = [250, 800];
[GNR.Sgt_spec,GNR.ks,GNR.tslide] = gabor_filt(S,Fs,dt,width,
    filtbound,pltfig);

% clean overtones
GNR.Sgt_thresh = GNR.Sgt_spec;
GNR.Sgt_thresh(GNR.Sgt_thresh < 7) = 0;

clear S y Fs

%% STEP 2: Read Pink Floyd

[y, Fs]      = audioread([datapath,musicfiles{2}]);
S            = y(1:length(y)-1)';
playsong     = 0;

if playsong == 1
    p8 = audioplayer(S,Fs);
        playblocking(p8);
end

%% STEP 2a: Filter Pink Floyd to find bass notes

width    = 100;
dt       = 0.5;
pltfig   = 0;
filtbound = [0, 250];

[PFb.Sgt_spec,PFb.ks,PFb.tslide] = gabor_filt(S,Fs,dt,width,
    filtbound,pltfig);

% clean overtones
PFb.Sgt_thresh = PFb.Sgt_spec/max(PFb.Sgt_spec,[],'all');
% Sgt_bassthresh(Sgt_bassthresh < 40) = 0;
PFb.Sgt_thresh(PFb.Sgt_thresh < 0.24) = 0;

%% STEP 2a: Filter Pink Floyd to find guitar notes

width    = 50;
dt       = 0.5;
pltfig   = 0;
filtbound = [250, 800];

```

```
[PFg.Sgt_spec,PFg.ks,PFg.tslide] = gabor_filt(S,Fs,dt,width,
    filtbound,pltfig);
clear S y Fs

PFg.Sgt_thresh = PFg.Sgt_spec/max(PFg.Sgt_spec,[],'all');
% Sgt_bassthresh(Sgt_bassthresh < 40) = 0;
PFg.Sgt_thresh(PFg.Sgt_thresh < 0.2) = 0;

%% STEP 3: Plot GNR

figure('units','inches','position',[1 1 10 16],'Color','w');
axes1 = axes('Position',[0.1 0.51 0.85 0.39]);
pcolor(GNR.tslide,GNR.ks,log(abs(GNR.Sgt_spec)+1).')
hold on
for i = 1:length(note.gHz)
    plot(GNR.tslide,repmat(note.gHz(i),[1, length(GNR.tslide)]),'k
        ')
end
plot(GNR.tslide,repmat(250,[1, length(GNR.tslide)]),'k')
plot(GNR.tslide,repmat(775,[1, length(GNR.tslide)]),'k')
shading flat
colormap(flipud(cmocean('grey')))
% box on
h1=gca;
set(h1,'tickdir','out','xminortick','on','yminortick','off');
set(h1,'ticklength',1.2*get(h1,'ticklength'));
set(h1,'fontsize',12);
ylabel('Musical Note','interpreter','latex','fontsize',20);
set(h1,'ytick',note.gHz,'yticklabel',note.glet);
set(h1,'xtick',[0:2:14],'xticklabel',{''});
xlim([min(GNR.tslide) max(GNR.tslide)])
ylim([250 775])
text(0,805,'(a) Guitar Spectrogram','interpreter','latex','
    fontsize',20);

axes2 = axes('Position',[0.1 0.06 0.85 0.39]);
pcolor(GNR.tslide,GNR.ks,log(abs(GNR.Sgt_thresh)+1).')
hold on
shading flat
for i = 1:length(note.gHz)
    plot(GNR.tslide,repmat(note.gHz(i),[1, length(GNR.tslide)]),'k
        ')
end
plot(GNR.tslide,repmat(250,[1, length(GNR.tslide)]),'k')
plot(GNR.tslide,repmat(775,[1, length(GNR.tslide)]),'k')
colormap(flipud(cmocean('grey')))
```

```
cb = colorbar('Position', [0.1 0.95 0.7 0.02],'Location','north');
% box on
h1=gca;
set(h1,'tickdir','out','xminortick','on','yminortick','off');
set(h1,'ticklength',1.2*get(h1,'ticklength'));
set(h1,'fontsize',12);
xlabel('time (sec)','interpreter','latex','fontsize',20);
ylabel('Musical Note','interpreter','latex','fontsize',20);
set(h1,'ytick',note.gHz,'yticklabel',note.glet);
set(h1,'xtick',[0:2:14],'xticklabel',{'0' '2' '4' '6' '8' '10' '12' '14'});
text(11.5,1458,'$\log(|s|+1)$','interpreter','latex','fontsize',20);
text(0,805,'(b) Cleaned Guitar Spectrogram','interpreter','latex','fontsize',20);
ylim([250 775])
Sname1 = [figfolder,'spectogram_GNR'];
print(Sname1,'-dpng')

%% STEP 4: Plot PF before filter

figure('units','inches','position',[1 1 10 16],'Color','w');
axes1 = axes('Position',[0.1 0.51 0.85 0.39]);
pcolor(PFg.tslide,PFg.ks,log(abs(PFg.Sgt_spec)+1).')
hold on
for i = 1:length(note.gHz)
    plot(PFg.tslide,repmat(note.gHz(i),[1, length(PFg.tslide)]),'k')
end
plot(PFg.tslide,repmat(250,[1, length(PFg.tslide)]),'k')
plot(PFg.tslide,repmat(775,[1, length(PFg.tslide)]),'k')
shading flat
colormap(flipud(cmocean('grey')))
% box on
h1=gca;
set(h1,'tickdir','out','xminortick','on','yminortick','off');
set(h1,'ticklength',1.2*get(h1,'ticklength'));
set(h1,'fontsize',14);
ylabel('Musical Note','interpreter','latex','fontsize',20);
set(h1,'ytick',note.gHz,'yticklabel',note.glet);
set(h1,'xtick',[0:10:60],'xticklabel',{''});
xlim([min(PFg.tslide) max(PFg.tslide)])
ylim([250 800])
text(50,880,'$\log(|s|+1)$','interpreter','latex','fontsize',20);
text(0,830,'(a) Guitar Spectrogram','interpreter','latex','fontsize',20);
```

```
axes2 = axes('Position',[0.1 0.06 0.85 0.39]);
pcolor(PFb.tslide,PFb.ks,log(abs(PFb.Sgt_spec)+1).')
hold on
shading flat
for i = 1:length(note.bHz)
    plot(PFb.tslide,repmat(note.bHz(i),[1, length(PFb.tslide)]),'k')
end
plot(PFb.tslide,repmat(250,[1, length(PFb.tslide)]),'k')
plot(PFb.tslide,repmat(775,[1, length(PFb.tslide)]),'k')
colormap(flipud(cmocean('grey')))
cb = colorbar('Position', [0.1 0.95 0.7 0.02],'Location','north');
% box on
h1=gca;
set(h1,'tickdir','out','xminortick','on','yminortick','off');
set(h1,'ticklength',1.2*get(h1,'ticklength'));
set(h1,'fontsize',14);
xlabel('time (sec)','interpreter','latex','fontsize',20);
ylabel('Musical Note','interpreter','latex','fontsize',20);
set(h1,'ytick',note.bHz,'yticklabel',note.blet);
set(h1,'xtick',[0:10:60],'xticklabel',{'0' '10' '20' '30' '40' '50' '60'});
text(0,144,'(b) Bass Spectrogram','interpreter','latex','fontsize',20);
ylim([65 140])
Sname1 = [figfolder,'spectogram_PF'];
print(Sname1,'-dpng')


%% STEP 5: Filtered

figure('units','inches','position',[1 1 10 16],'Color','w');
axes1 = axes('Position',[0.1 0.51 0.85 0.39]);
pcolor(PFg.tslide,PFg.ks,log(abs(PFg.Sgt_thresh)+1).')
hold on
for i = 1:length(note.gHz)
    plot(PFg.tslide,repmat(note.gHz(i),[1, length(PFg.tslide)]),'k')
end
plot(PFg.tslide,repmat(250,[1, length(PFg.tslide)]),'k')
plot(PFg.tslide,repmat(775,[1, length(PFg.tslide)]),'k')
shading flat
colormap(flipud(cmocean('grey')))
% box on
h1=gca;
```

```
set(h1,'tickdir','out','xminortick','on','yminortick','off');
set(h1,'ticklength',1.2*get(h1,'ticklength'));
set(h1,'fontsize',14);
ylabel('Musical Note','interpreter','latex','fontsize',20);
set(h1,'ytick',note.gHz,'yticklabel',note.glet);
set(h1,'xtick',[0:10:60],'xticklabel',{''});
xlim([min(PFg.tslide) max(PFg.tslide)])
ylim([250 800])
% text(50,850,'$\log(|s|+1)$','interpreter','latex','fontsize',20)
    ;
text(0,830,'(a) Cleaned Guitar Spectrogram','interpreter','latex',
    'fontsize',20);
caxis([0 .2])

axes2 = axes('Position',[0.1 0.06 0.85 0.39]);
pcolor(PFb.tslide,PFb.ks,log(abs(PFb.Sgt_thresh)+1).')
hold on
shading flat
for i = 1:length(note.bHz)
    plot(PFb.tslide,repmat(note.bHz(i),[1, length(PFb.tslide)]),'k
        ')
end
plot(PFb.tslide,repmat(250,[1, length(PFb.tslide)]),'k')
plot(PFb.tslide,repmat(775,[1, length(PFb.tslide)]),'k')
colormap(flipud(cmocean('grey')))
% cb = colorbar('Position', [0.1 0.95 0.7 0.02],'Location','north
    ');
% box on
h1=gca;
set(h1,'tickdir','out','xminortick','on','yminortick','off');
set(h1,'ticklength',1.2*get(h1,'ticklength'));
set(h1,'fontsize',14);
xlabel('time (sec)','interpreter','latex','fontsize',20);
ylabel('Musical Note','interpreter','latex','fontsize',20);
set(h1,'ytick',note.bHz,'yticklabel',note.blet);
set(h1,'xtick',[0:10:60],'xticklabel',{'0' '10' '20' '30' '40' '50
    ' '60'});
text(0,144,'(b) Cleaned Bass Spectrogram','interpreter','latex','
    fontsize',20);
ylim([65 140])
caxis([0 .2])
Sname1 = [figfolder,'spectogram_PF_cleaned'];
print(Sname1,'-dpng')

%% Example gabor transform spectra

```

```
[y, Fs]      = audioread([datapath,musicfiles{1}]);
S            = y';
playsong     = 0;

figure('units','inches','position',[1 1 10 14],'Color','w');
L         = length(S)/Fs; % record time in seconds
n         = length(S);
t2        = linspace(0,L,n+1);
t         = t2(1:n);
k         = (1/L)*[0:n/2-1 -n/2:-1];
ks        = fftshift(k);
tslide    = 0:dt:L;
width     = 10;
g         = exp(-width*(t-L/2).^2);
Sg        = g.*S;
Sgt       = fft(Sg);

subplot(3,1,1)
plot(t,S,'k')
hold on
plot(t,g,'k','Linewidth',[2])
set(gca,'Fontsize',14)
ylabel('$S(t), g(t)$','interpreter','latex','fontsize',20)
xlabel('time (t)','interpreter','latex','fontsize',20)
h1=gca;
set(h1,'tickdir','out','xminortick','on','yminortick','off');
set(h1,'ticklength',1.2*get(h1,'ticklength'));
text(0.2,0.9,'(a) Timeseries, Gabor Window','interpreter','latex','fontsize',18);

subplot(3,1,2)
plot(t,Sg,'k')
set(gca,'Fontsize',14)
ylabel('$S(t)g(t)$','interpreter','latex','fontsize',20)
xlabel('time (t)','interpreter','latex','fontsize',20)
h1=gca;
set(h1,'tickdir','out','xminortick','on','yminortick','off');
set(h1,'ticklength',1.2*get(h1,'ticklength'));
text(0.2,0.032,'(b) Filtered Data','interpreter','latex','fontsize',18);

subplot(3,1,3)
plot(ks,abs(fftshift(Sgt))/max(abs(Sgt)),'k')
% axis([-50 50 0 1])
set(gca,'Fontsize',14)
ylabel('FFT($Sg$)','interpreter','latex','fontsize',20)
```

```
xlabel('frequency ($\omega$)','interpreter','latex','fontsize',20)
h1=gca;
set(h1,'tickdir','out','xminortick','on','yminortick','off');
set(h1,'ticklength',1.2*get(h1,'ticklength'));
text(-2.4*10^4,0.9,'(c) Spectra of Gabor Window','interpreter','
    latex','fontsize',18);

Sname1 = [figfolder,'gabor_ex'];
print(Sname1,'-dpng')
```

References

J Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.