AMATH 482 Homework 2

Sophia Jannetty

February 10, 2021

Abstract

Here, I use the Gabor Transform as a tool to identify the central frequencies (or notes) being played by the guitar in the introduction to the Guns and Roses song "Sweet Child O' Mine", and by both the bass and solo guitar during an excerpt of the Pink Floyd song "Comfortably Numb". For each song I first eliminate all frequencies irrelevant to my instrument of interest, then use the Gabor transform followed by additional Gaussian filtering in the frequency domain to identify the central frequencies being emitted by my instruments of interest at each time point. From these data I construct spectrograms, revealing the notes each instrument was playing at each time point. Notably my efforts to isolate the guitar solo notes from "Comfortably Numb" were largely unsuccessful. This document outlines the theoretical basis for the time frequency analysis employed and provides details pertaining to my MATLAB implementation of this analysis. My intended audience is my graders and my future self.

1 Introduction and Overview

The Gabor Transform is a time-frequency analysis method used to determine the magnitudes of a signal's constituent frequencies at a representative series of time windows throughout a signal. This strategy allows us to identify both the frequencies present in the signal and when those frequencies were present. In this exercise, I use the Gabor Transform to identify which notes an instrument was playing at what times during excerpts from two classic rock songs. Because the Gabor Transform returns all the frequencies present in a certain time window, my first step in isolating each instrument was using a bandpass filter to eliminate frequencies from the signal that I knew were not relevant to the instrument of interest. I then Gabor transformed each signal to identify what remaining frequencies were present during a representative series of time windows. Based on a few assumptions, I identify the central frequency each instrument is emitting in each time window and apply a Gaussian filter centered at this frequency. For the guitar solo in "Comfortably Numb", I also applied upside-down Gaussian filters at three frequencies in my frequency range that I expected might be played by other instruments to minimize the magnitudes of a few extraneous frequencies before identifying the central frequencies emitted by the guitar. I used these processed data to construct spectrograms depicting which notes the instruments were playing when. This technique worked reasonably well for isolating the guitar part in "Sweet Child O' Mine" and the bass part in "Comfortably Numb", but did not work very well for isolating the solo guitar in "Comfortably Numb".

2 Theoretical Background

2.1 The Discrete and Fast Fourier Transform

The Discrete Fourier Transform (DFT) takes a vector of N data points and returns a vector given by formula 1.

$$\hat{x}_k = \frac{1}{N}\sum_{n=0}^{N-1} x_n e^{\frac{2\pi i k n}{N}} \tag{1}$$

This tests for the presence of the specified set of frequencies given by $k = 0, 1, 2, ..., N-1$. By capping the maximum wave number at $N-1$, the discrete Fourier transform is taking into account the fact that it is unable to detect frequencies that are so high that oscillations would happen between our sampled points. Each $\hat{x}_k$ value returned is a Fourier coefficient, or a measure of the magnitude of frequency k in the function underlying the input data.

Though the function underlying the input data is real, the coefficients can be complex numbers with the

real part of the number corresponding to the even components of the function and the imaginary part of the number corresponding to the odd components of the function[1]. If f(x) is a real-valued function (as is the case with this assignment), then c_0 is real and the coefficients are symmetrical around zero ($c_k = c_{-k}$). The Fast Fourier transform (FFT) is a Discrete Fourier Transform algorithm that runs in $O(n\log(n))$ operations instead of $O(n^2)$ operations [1].

The Fourier Transform is a powerful tool for quantifying what frequencies are present in a signal, but it can not tell us which frequencies happened when in the signal. Fourier transforming a function requires integrating out time, so we lose the ability to localize specific frequencies in time. If we want information regarding which frequencies were present at different time points throughout the signal, we need to use a time-frequency analysis technique like the Gabor Transform.

2.2 Time-Frequency Analysis: The Gabor Transform

2.2.1 Gabor Conceptual Overview

To take the Gabor Transform of a signal, multiply the signal in the time domain by a filter function of width a shifted to time τ. This filter function must be real, symmetric, and have an L_2 norm set to unity. Applying a filter function that satisfies these requirements maximizes the magnitudes of the frequencies in the window encompassed by the filter and minimizes the magnitudes of all other frequencies. The Gaussian is commonly used as a filter function and is used in this project. Once this filter has been applied, the newly scaled function is FFT'd and all frequencies present are assumed to be encompassed by the filter function and thus present at time τ. The τ is then moved progressively down the domain of the signal and the constituent frequencies at each τ (and relative powers of these constituent frequencies) are calculated.

The width of the Gabor window a (or variance of the Gaussian when using a Gaussian as your filter function) is an important parameter to optimize. Any frequencies with wavelengths longer than the window will be completely lost. However, if the window is too wide, our analysis will tell us that frequencies that happened significantly before or after time τ happened at time τ, so we will lose localization of our frequencies in time. This trade-off between frequency and time localization is the Heisenberg uncertainty principal. The width of the Gabor window must be set to a width that strikes an acceptable balance given the analysis task at hand.

2.2.2 Discrete Gabor Transform

In the discrete Gabor Transform, the filter function is shifted down the domain of the signal to a defined set of time points and the constituent frequencies at each of these discrete τ values (and relative powers of these constituent frequencies) are calculated. The formula for these τ values is $k = m\omega_0$ where m is an integer and ω_0 is a positive constant representing the frequency resolution. Similarly only a discrete set of frequencies is considered, as described in by the equation $\tau = nt_0$. In this equation, n is an integer and t_0 is the increment by which τ will change with each step. The discrete Gabor transform is given by equation 2

$$\tilde{f}_g(m,n) = \int_{-\infty}^{\infty} f(t)g(t-nt_0)e^{2\pi i m\omega_0 t}dt \tag{2}$$

Because the window size and τ are both discretely defined, it is important to remember that in order to accurately analyze all frequencies at all time points there must be a bit of overlap of the window between one τ value and the next. Thus the window must be defined to be sufficiently wide or the t_0 steps sufficiently small to get overlap. Similarly, the frequency resolution ω_0 must be sufficiently small to capture all frequencies of interest (any frequencies present in the signal with a shorter wavelength than ω_0 will be too fast for the analysis to reliably detect, which will result in aliasing).

2.3 Filtering in the Frequency Domain

2.3.1 Bandpass Filtering

Each instrument part I was interested in isolating spanned only a subset the of frequencies present in the song. I removed all frequencies I knew would be irrelevant to my instrument of interest by setting the coefficients of all frequencies outside a determined relevant range to zero. I did this prior to my Gabor filter application. Relevant ranges were determined by listening to the music and identifying the range of notes spanned by each instrument part.

2.3.2 Gaussian Filtering to Eliminate Overtones

The convolution theorem states that multiplication in the frequency domain results in convolution in the space domain [1]. At each discrete τ value, the song was Gabor transformed in the time domain and then Fast Fourier Transformed, allowing for a quantification of the magnitudes of the song's constituent frequencies at that time. I then multiply these frequency domain data by a Gaussian filter shifted to be centered on the central frequency my instrument of interest is emitting at this time point. This maximizes frequencies at and around the central frequencies while minimizing or deleting all other frequencies. My method for identifying this central frequency is outlined in the Algorithm Implementation and Development section.

I also used Gaussian filtering when analyzing the guitar solo in "Comfortably Numb" to minimize notes that I expected to be present at a given time but that were not played by the solo guitar. When trying to eliminate specific notes from the signal, I multiplied the signal by 1−(an appropriately sized Gaussian) shifted to be centered at the frequencies of notes I wanted eliminated. This minimized or eliminated notes at and immediately around the specified frequencies while multiplying the rest of the frequencies in the signal by 1, leaving them unchanged.

2.4 Spectrograms and Musical Notes

Spectrograms have one column per τ value shifted to while Gabor transforming. Along the Y axis are all of the frequencies tested for at each time slice and the frequencies are colored like a heat map based on whether the coefficients for those frequencies are high or low at each τ. Musical notes are names for sound waves at defined frequencies. I labeled relevant note frequencies as horizontal lines at the correct Y locations on the spectrograms I created for this project.

3 Algorithm Implementation and Development

For a conceptual overview of the algorithm I used for my analysis, please see Algorithm 1. Note that this overview omits initial data processing steps required when analyzing "Comfortably Numb" (which are outlined in the Initial Setup subsection). The overview includes steps taken exclusively when analyzing the guitar solo in "Comfortable Numb" in italics.

Algorithm 1: Overview of Analysis Algorithm.
Note: *Italicized* steps only performed when analyzing guitar solo part in "Comfortably Numb"

Import song to vector y using `audioread`
BANDPASS FILTERING:
Calculate Discrete Fourier Transform of y
Filter by setting all coefficients of frequencies outside defined frequency range to zero
Inverse Fourier Transform filtered song and save in vector fy
GABOR TRANSFORM:
for Every Defined τ value **do**
  Create Gaussian filter vector `g` shifted to be centered at τ
  Multiply `g` by `fy`, save in vector `fyg`
  Calculate Discrete Fourier Transform of `fyg` and save in vector `fygt`
  ADDITIONAL GAUSSIAN FILTERING:
  Define three (1 - Gaussian) filters and shift to be centered at three expected overtones
  Multiply `fygt` by three shifted upside-down Gaussian filters
  Find set `s` of five frequencies with greatest Fourier coefficients (highest power)
  Define Gaussian filter `f` centered at the lowest frequency in set `s`
  Multiply `fygt` by `f` and save product in matrix `spec`
end for
Use `spec` data and `pcolor` function to plot spectrogram

3.1 Initial Setup

For each song my initial steps involved importing the song excerpt and setting up the discretized time and frequency components needed to Fast Fourier Transform the signal.

For "Sweet Child O' Mine", I used the `audioread` function to import the song excerpt into vector `y` and to extract the sampling rate (in Hz) at which the song was recorded in `Fs`. I calculated the length of the song excerpt in seconds by dividing the number of measurements in `y` by this sampling rate. I set my time domain `L` to be equal to this length in seconds and the number of frequency components (or the spectral resolution) `n` to be equal to the number of samples in vector `y`. I discretized the time domain by using the `linspace` function to create a linearly spaced vector of `n`+1 points spanning from 0 to L and then deleting the last point. I delete this last point because the fast Fourier transform assumes the function underlying my data is periodic so the first point would be the same as the last. I then defined my frequency components as a vector of `k` frequencies. In order to have my frequency components defined in oscillations per second (Hz) as opposed to oscillations per duration-of-excerpt `L`, I multiplied my domain by $\frac{1}{\mathtt{L}}$. I then defined my frequencies to count from 0 to $\frac{\mathtt{n}}{2} - 1$ and then from $-\frac{\mathtt{n}}{2}$ to -1 in order to mimic the order in which data is returns from MATLAB's FFT function. When I run MATLAB's FFT function on my song vector `y`, it will return a vector of coefficients that corresponds to the frequency held in the same index location in my `k` vector.

The "Comfortably Numb" excerpt had an odd number of data points, making it impossible to match its dimensions with a frequency array (the frequency array must count from 0 to $\frac{n}{2} - 1$ and then from -$\frac{n}{2}$ to -1, thus n must be an even number). To address this I deleted the last data point of the vector, resulting in a slight loss of information. This clip was also very long. To improve the runtime of my algorithm, I split the clip into quarters and performed each step of this analysis on each quarter individually. I stored these quarters in a struct. Because I only worked with one quarter at a time, I defined my time domain `L` to be equal to the length of a quarter of the clip in seconds and defined the number of frequency components `n` to be the number of samples in each quarter (or $\frac{1}{4}$ the total number of samples in the excerpt). Using these values I followed the same subsequent setups steps as outlined above. For the remainder of this document when speaking about performing an analysis step on "Comfortably Numb", know that I iterated through the four quarters of the song and performed each step of the analysis on each quarter individually.

3.2 Bandpass Filtering

I determined the range of frequencies I would expect each instrument to be emitting throughout each song except by determining what I thought were the lowest and highest note each instrument played in the excerpt. For "Sweet Child of Mine" my range was 200-800 Hz, for the Bass in "Comfortably Numb" my range was 0-140 Hz (I assumed the Bass was emitting the lowest frequencies of any instrument in the recording) and for the guitar in "Comfortably Numb" my range was 440-1100 Hz. I FFTed each signal and multiplied it in the frequency domain by a binary filter that has a 1 at every index that's frequency (found in the corresponding index in my `k` vector) is within my defined range and a 0 at all other frequencies. This deletes all frequencies from my signal that I know are irrelevant to my instruments of interest. I then IFFTed this filtered signal back into the time domain and listened to the resulting audio clips to see if I felt the range appropriately isolated the instrument part of interest.

3.3 Gabor Transform

I used a Gaussian as my filter function. For both guitar parts I set the width `a` of my Gaussian to be 200, but when extracting the bass I set `a` to 10 to accommodate the bass's longer wavelengths (the variance of the Gaussian is $\frac{1}{\mathtt{a}}$, so a smaller `a` results in a wider Gaussian). I determined these widths through a process of trying different widths, looking at the resulting spectrograms, and adjusting until I reached an acceptable balance of frequency localization and time localization.

Next I defined by set of τ values. For "Sweet Child O' Mine" the guitar played notes of the same length equally spaced at the same pace throughout the recording. I was able to count the number of notes played (57) and assumed that one note was played every $\frac{\mathtt{L}}{57}$ seconds. By shifting the window in intervals of $\frac{\mathtt{L}}{2*57}$ I determined I would center the Gaussian of my Gabor transform on each played note twice, ensuring each note played by the guitar would be captured in my transform. I set my series `tslide` of τ values to count from 0 to `L` in increments of $\frac{L}{2*57}$. The rhythms in "Comfortably Numb" were more complex. For both the guitar and the bass I set my series `tslide` of τ values to count from 0 to `L` in increments of .2. The pace of

"Comfortably Numb" is about 64 beats per minute, so by incriminating by .2 I am sampling slightly over 4 times per measure. This should be sufficient to capture all the Bass notes but may not be small enough to accurately capture some of the shorter notes played by the guitar solo (see conclusion section).

For each bandpass-filtered signal, I iterated through my `tslide` vector and constructed a Gaussian of width `a` shifted to be centered at each τ value I. I then multiplied this Gaussian by my signal data in the time domain, maximizing the magnitudes of frequencies encompassed by the Gaussian and minimizing all others. I then FFT'd the data to quantify the magnitudes of the song's constituent frequencies at this time point.

3.4 Additional Gaussian Filtering in the Frequency Domain

3.4.1 Filtering to Amplify the Central Frequency

At this point there were still a lot of frequencies present in the signal that were not being emitted by my instrument of interest. I next tried to identify the central frequency being emitted by each instrument at each τ. My method relied on two assumptions. One was that the central frequency the instrument of interest was emitting would be one of the five maximum magnitude frequencies at this time point. The second was that the other four maximum magnitude frequencies would be overtones or other higher-pitch notes. I found the maximum five coefficients at each time point and considered the lowest frequency note of that set to be the central frequency. I then constructed a Gaussian filter in the frequency domain (using my frequency vector k) of width `a2` = .2 and shifted it to be centered at this central frequency. I multiplied my signal by this Gaussian, maximizing the coefficients of the frequencies at and immediately around the central frequency while minimizing all other frequencies. My filter width was determined through trial and error; I choose a width that upon visual inspection yielded a satisfactorily filtered signal.

3.4.2 Filtering to Eliminate Specific Frequencies

For the guitar solo of "Comfortably Numb" many large-magnitude frequencies were present at lower frequencies than the central frequencies emitted by the guitar. I attempted to address this issue by removing three frequencies that I expected would be present in the signal before identifying the central frequency at each time point as described above. Because I knew that overtones from the bass would be present in my frequency range and it sounded like the rhythm guitar was playing notes similar to those played by the bass, I decided to remove three overtones of the frequency the bass was emitting at the τ in question. I ran my analysis of the bass part first and kept track of the central frequency the bass was emitting at each τ in vector `cent_freq_vals` . During my analysis for the guitar, at each τ I calculated the three frequencies representing the pitches one, two, and three octaves above the pitch played by the bass at that τ. I then created three upside-down Gaussian filters calculated by subtracting the formula for a Gaussian centered at the frequencies I wanted to eliminate from 1. I set the width of this Gaussian `a3` to .2. I determined this width through trial and error. I multiplied my signal vector (in the frequency domain) by each of these filters, minimizing the frequencies at and around this specified set of three overtones while leaving all other frequencies unchanged, before finding the central frequencies emitted by the guitar as described in section 3.4.1.

3.5 Spectrogram Construction

I saved the vector of Fourier coefficients at each τ in a matrix `spec` and used the `pcolor` function to construct spectrograms of each instrument part. Because the function underlying the songs are real valued, the coefficient values are symmetric around zero so I only plotted the positive frequencies. Because the coefficients are complex numbers I plotted the absolute value of the coefficients to capture the magnitude of the imaginary part of the number. Also, because my filtering made the coefficients vary by such great margins, I plotted the $log(abs(\texttt{spec})+1)$. I then looked at the central frequencies emitted at each time point, found the musical notes closest to those frequencies, and plotted labeled horizontal lines at those relevant frequencies.

4 Computational Results

Representative spectrograms for each instrument part isolated can be seen in Figure 1. The bass and guitar spectrograms for "Comfortably Numb" depict only the first quarter of the excerpt. Full bass and

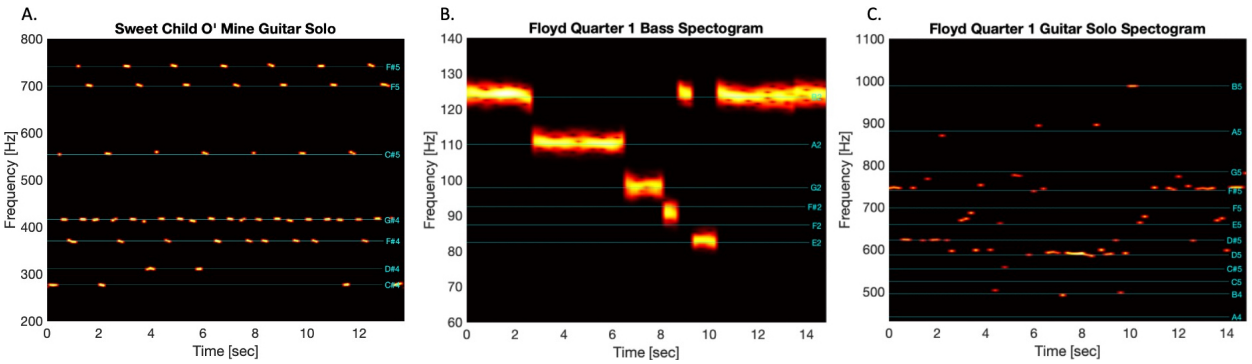

Figure 1: Spectrograms depicting notes played by guitar in Sweet Child O' Mine (A), and both bass (B) and solo guitar (C) in the first quarter of "Comfortably Numb". Relevant frequencies are labeled with their corresponding musical notes. Full spectograms can be found in Appendix C. Notably, the bass line repeated the above depicted pattern in each quarter.

guitar spectrograms for "Comfortably Numb" and a larger spectrogram for "Sweet Child O' Mine" can be found in Appendix C. The notes played by each instrument at each time point can be found by identifying the frequency of the brightest point in the relevant spectrogram at the time of interest. Accuracy of spectrograms was assessed by comparing found notes to the notes played in the excerpts by ear.

5 Summary and Conclusions

This analysis method was successful at isolating the guitar introduction to "Sweet Child O' Mine". This excerpt was ideal for this analysis because for the most part the guitar was playing louder than all the other instruments (when there were other instruments playing at all). Additionally all other powerful frequencies measured were higher than the guitar's central frequency, so my assumptions when finding the central frequency at each τ held true. The consistent rhythm of the guitar part meant I was confident my `tslide` vector contained a representative set of time points to analyze.

This analysis was also successful at isolating the bass part in the "Comfortably Numb" excerpt. The bass was the emitting the lowest frequencies in the signal so again my guiding assumption that the central frequency will be the lowest of high-magnitude frequencies held true. Additionally the Bass was playing a fairly simple rhythm, though my analysis may not have accurately captured the timing of each note played.

This analysis was unsuccessful when isolating the guitar solo in "Comfortably Numb". There were many other instruments emitting powerful frequencies within the guitar's frequency range during the solo. My assumption that the central frequency was the lowest of the high-magnitude frequencies likely was not true at many time points. My attempt to remedy this by filtering out specific bass overtone frequencies was not sufficient to remove all non-guitar frequencies present at each slice. Furthermore there were likely times when the guitar was playing the same note as the bass and I accidentally filtered it out. Additionally, my τ values only iterated to about four time points per measure, so it is possible the guitar played multiple notes within each filter window. By selecting one frequency to filter around I likely eliminated many shorter notes. To improve this analysis I could decrease my τ stepping increment and redefine my band-pass filter at each τ to be narrowly centered around the note the guitar is playing (which I can determine by listening to the song), but at that point I might as well just reconstruct the guitar part by ear.

The Gabor transform is a useful time-frequency analysis technique, but it is limited in its ability to localize in both time and frequency concurrently. Prioritizing localization in one domain requires sacrificing localization in the other. In order to get excellent localization in both domains, a more advanced time-frequency analysis like the Wavelet Transform must be employed.

References

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

Appendix A MATLAB Functions

Table 1: Notable Matlab Functions

`[y, Fs] = audioread('file.m4a')`	reads data from file and returns sampled data y, and a sample rate for that data Fs.
`y = linspace(x1,x2,n)`	returns a row vector of `n` evenly spaced points between `x1` and `x2`
`fygt = fft(fyg);`	MATLAB's FFT implementation for one dimensional data. Returns vector `fygt` of Fourier coefficients that describe the function underlying the data in vector `fyg`. Frequencies are ordered as described in `ks = fftshift(k)` above. Each coefficient's frequency can be found in at that coefficient's index in the frequency vector `k`
`ks = fftshift(k)`	used to reorder wavenumbers to match output structure of `fftx()`, which returns wavenumbers from 0 to $\frac{n}{2}-1$, and then $\frac{n}{2}$ to -1. Allows for wavenumber axis to be continuous from $\frac{-n}{2}$ to $\frac{n}{2}$ while plotting.
`fy = ifft(sy)`	MATLAB's inverse FFT implementation for one dimensional data. Takes the vector `sy` of Fourier coefficients and returns vector `fy` of data returned to the time domain.
`[max_vals, idxs] = maxk(mfygt, 5)`	returns five max point from vector `mfygt` in vector `max_vals` and their corresponding index locations in `mfygt` in vector `idxs`. Used when finding central frequencies in each τ window.
`central_freq = min(idxs)`	saves the minimum value in vector `idxs` to variable `central_freq`. Used when finding central frequencies in each τ window.
`pcolor[X, Y, C]`	Used to create spectrograms. Makes a heat map of specified X and Y values with magnitudes as specified in matrix C. C's dimensions must be length(X) by length(Y).

Appendix B MATLAB Code

```
%% Guns and Roses Guitar
clear all; close all; clc
[y, Fs] = audioread('GNR.m4a'); %Fs is sampling rate
tr_gnr = length(y)/Fs; % record time in seconds

%% Fourier Transform of whole song
L = tr_gnr; n = length(y); %L is time in seconds rounded up
%n isn't power of 2 -> relying on matlab to pad
%make linear vector of n points spanning time domain
t2 = linspace(0, L, n+1);
t = t2(1:n); %matlab fft assumes periodic domain so last point = first
%make frequency vector in Hz and shift for plotting
k = (1/L) * [0:n/2-1 -n/2:-1]; ks = fftshift(k);
sy = fft(y);
%% Bandpass Filter
% remove all wavenumbers higher and lower than guitar range
binary_filter = abs(k()) >= 800;
sy(binary_filter) = 0;
binary_filter = abs(k()) <= 200;
sy(binary_filter) = 0;
fy = ifft(sy);

%% Gabor Transform
a = 200; %width of gaussian, 1/a = variance of gaussian
fygt_spec=[]; %matrix for collecting spectrogram data

% Define tau vector
increment = tr_gnr/(2*57); %57 notes in section, sample twice per note
tslide=0:increment:tr_gnr;

for j = 1:length(tslide)
    g = exp(-a *(t-(tslide(j))).^2); %Define Gaussian shifted to tau
    %Apply Gaussian in Time Domain
    fyg = g'.*fy; %Invert filter so dimensions match data
    fygt = fft(fyg);

    %filter around central frequency
    mfygt = fygt;
    [max_vals, idxs] = maxk(mfygt, 5); %find 5 max power points
    central_freq = min(idxs); %assume central frequency is longest
                              %wavelength of these

    % Make Gaussian Filter centered at central frequency
    a2 = .2;
    filter = (exp(-a2* ((k-k(central_freq)) .^2)));
    mfygt = mfygt'.*filter; %Apply Gaussian
    fygt_spec(:,j) = fftshift(abs(mfygt));
end
%% Plot
figure()
pcolor(tslide, ks, log(abs(fygt_spec)+1)), shading interp
set(gca, 'Ylim',[200, 800], 'Fontsize', 16)
```

```
colormap(hot)
xlabel('Time [sec]'); ylabel('Frequency [Hz]');
title('Sweet Child O'' Mine Guitar Solo');

%Label relevant frequencies
cl = yline(277.183,'-','C#4','LineWidth',.5);
cl.LabelVerticalAlignment = 'middle';
cl.Color = 'cyan';
ccl = yline(554.365,'-','C#5','LineWidth',.5);
ccl.LabelVerticalAlignment = 'middle';
ccl.Color = 'cyan';
cl.Color = 'cyan';
gl = yline(415.305,'-','G#4','LineWidth',.5);
gl.LabelVerticalAlignment = 'middle';
gl.Color = 'cyan';
dl = yline(311.127,'-','D#4','LineWidth',.5);
dl.LabelVerticalAlignment = 'middle';
dl.Color = 'cyan';
ddl = yline(698.456,'-','F5','LineWidth',.5);
ddl.LabelVerticalAlignment = 'middle';
ddl.Color = 'cyan';
ffl = yline(739.989,'-','F#5','LineWidth',.5);
ffl.LabelVerticalAlignment = 'middle';
ffl.Color = 'cyan';
ffl = yline(369.994,'-','F#4','LineWidth',.5);
ffl.LabelVerticalAlignment = 'middle';
ffl.Color = 'cyan';

%% Pink Floyd Bass
clear all; close all; clc

[y, Fs] = audioread('Floyd.m4a'); %Fs is sampling rate
y = y(1:(length(y)-1));% Subtract one because number of samples is odd
tr_gnr = length(y)/Fs; % record time in seconds

%% Split song into quarters, 4 4-measure sections, save in struct
fourth = (length(y) / 4);
full_song_1 = y(1:fourth);
full_song_2 = y(fourth+1:(2*fourth));
full_song_3 = y((2*fourth)+1:(3*fourth));
full_song_4 = y((3*fourth)+1:end);
B_full_song = struct('one', full_song_1, 'two', full_song_2, 'three', ...
    full_song_3, 'four', full_song_4);
tr_gnr = fourth/Fs; %length of each quarter in seconds
song_clips = fieldnames(B_full_song); %used to iterate through quarters

%% Bandpass Filter
L = tr_gnr; n = fourth; %L=length of each quarter in seconds
t2 = linspace(0, L, n+1); %linear vector of n points
t = t2(1:n);%matlab fft assumes periodic domain so last point = first
%make frequency vector in Hz and shift for plotting
k = (1/L) * [0:n/2-1 -n/2:-1]; ks = fftshift(k);

for i = 1:numel(song_clips) %iterate through each quarter of the song
```

```
    sy = fft(B_full_song.(song_clips{i}));
    % Threshold Filter by Frequency
    % remove all wavenumbers higher than my bass
    binary_filter = abs(k()) >= 140;
    sy(binary_filter) = 0;
    B_full_song.(song_clips{i}) = ifft(sy);
end

%% Gabor Transform
a = 10; %width of gaussian for gabor

%matrices for holding spectrogram data
bspecs = {'bspec1', 'bspec2', 'bspec3', 'bspec4'};
%matrices for holding central frequency data
cent_freq_vals = {'cfv1', 'cfv2', 'cfv3', 'cfv4'};

for i = 1:numel(bspecs) %iterate through each quarter of the song
    %Gabor filter
    fygt_spec=[]; %spectrogram matrix
    cfvs=[]; %central frequency matrix
    % Define tau vector
    tslide=0:.2:tr_gnr;
    fy = B_full_song.(song_clips{i}); %extract y vec for this quarter
    for j = 1:length(tslide)
        g = exp(-a *(t-(tslide(j))).^2); %Define Gaussian shifted to tau
        fyg = g'.*fy; %Apply in time domain
        mfygt = fft(fyg);

        %filter by power, gaussian in frequency domain centered at max power
        %frequency
        [max_vals, idxs] = maxk(mfygt, 5); %find 5 max power points
        central_freq = min(idxs);
        cfvs(j) = k(central_freq);


        % Make Gaussian Filter
        a2 = .2;
        % Shift to central frequency
        filter = (exp(-a2* ((k-k(central_freq)) .^2)));
        mfygt = mfygt'.*filter; %apply Gaussian

        fygt_spec(:,j) = fftshift(abs(mfygt)); %shift for plotting
    end
    B_full_song.(bspecs{i}) = fygt_spec; %save coefficients
    B_full_song.(cent_freq_vals{i}) = cfvs; %save central frequencies
end
%% Plot

for i = 1:numel(bspecs) %make a different plot for each quarter
    figure()
    fygt_spec = B_full_song.(bspecs{i});
    pcolor(tslide, ks, log(abs(fygt_spec)+1)), shading interp
    set(gca, 'Ylim',[60, 140], 'Fontsize', 16)
    colormap(hot)
```

```
161
162         xlabel('Time [sec]'); ylabel('Frequency [Hz]');
163         title_str = sprintf('Floyd Quarter %d Bass Spectogram', i);
164         title(title_str);
165
166         %Label Relevant Notes
167
168         bl = yline(123.471,'-','B2','LineWidth',.5);
169         bl.LabelVerticalAlignment = 'middle';
170         bl.Color = 'cyan';
171
172         al = yline(110,'-','A2','LineWidth',.5);
173         al.LabelVerticalAlignment = 'middle';
174         al.Color = 'cyan';
175
176         gl = yline(97.99,'-','G2','LineWidth',.5);
177         gl.LabelVerticalAlignment = 'middle';
178         gl.Color = 'cyan';
179
180         el = yline(82.407,'-','E2','LineWidth',.5);
181         el.LabelVerticalAlignment = 'middle';
182         el.Color = 'cyan';
183
184         fl = yline(92.499,'-','F#2','LineWidth',.5);
185         fl.LabelVerticalAlignment = 'middle';
186         fl.Color = 'cyan';
187
188         fl = yline(87.307,'-','F2','LineWidth',.5);
189         fl.LabelVerticalAlignment = 'middle';
190         fl.Color = 'cyan';
191 end
192
193 %% Pink Floyd Guitar
194 [y, Fs] = audioread('Floyd.m4a'); %Fs is sampling rate
195 y = y(1:(length(y)-1)); %make n even
196 tr_gnr = length(y)/Fs; % record time of full clip in seconds
197
198 %% Split song into fourths
199 fourth = (length(y) / 4);
200 full_song_1 = y(1:fourth);
201 full_song_2 = y(fourth+1:(2*fourth));
202 full_song_3 = y((2*fourth)+1:(3*fourth));
203 full_song_4 = y((3*fourth)+1:end);
204 G_full_song = struct('one', full_song_1, 'two', full_song_2, 'three', ...
205     full_song_3, 'four', full_song_4);
206 tr_gnr = fourth/Fs; %time of each quarter in seconds
207 song_clips = fieldnames(G_full_song); %used for iterating
208
209 %% Bandpass Filter
210 %make initial vectors to be used for whole thing
211 L = tr_gnr; n = fourth; %L=length of each quarter in seconds
212 %make linear vector of n points spanning duration of each quarter
213 t2 = linspace(0, L, n+1);
214 t = t2(1:n);
```

```
k = (1/L) * [0:n/2-1 -n/2:-1]; ks = fftshift(k); %make wave number vector and
    shifted for plotting

for i = 1:numel(song_clips)%iterate through each quarter of the song
    sy = fft(G_full_song.(song_clips{i}));
    % Filter by Frequency
    minfreq = 440;
    maxfreq = 1100;
    binary_filter = abs(k()) >= maxfreq;
    sy(binary_filter) = 0;
    binary_filter = abs(k()) <= minfreq;
    sy(binary_filter) = 0;
    G_full_song.(song_clips{i}) = ifft(sy);
end

%% Gabor Transform
a = 200; %width of gaussian for gabor

gspecs = {'gspec1', 'gspec2', 'gspec3', 'gspec4'};
cent_freq_vals = {'cfv1', 'cfv2', 'cfv3', 'cfv4'};

for i = 1:numel(gspecs)%iterate through each quarter of the song
    fygt_spec=[];
    cfvs=[];
    tslide=0:.2:tr_gnr;
    fy = G_full_song.(song_clips{i}); %extract y vec for this quarter
    %Extract bass notes to filter out bass overtones
    bass_central_freqs = B_full_song.(cent_freq_vals{i});
    for j = 1:length(tslide)
        g = exp(-a *(t-(tslide(j))).^2); %Define Gaussian shifted to tau
        fyg = g'.*fy; %Apply Gaussian in time domain
        mfygt = fft(fyg);

        %first inverse gaussian around notes octaves up from central
        %frequency emitted by bass at this time point (to filter out
        %rhythm guitar that is playing these notes)
        a3 = .2; %width of this gaussian
        bass_note = bass_central_freqs(j); %bass note out of relevant range
        bass_note2 = 2 * bass_note; %1 octave up
        bass_note3 = 2 * bass_note2; %2 octaves up
        bass_note4 = 2 * bass_note3; %3 octaves up
        %Define upside-down Gaussians centered at each of these frequencies
        guitar_rm_filter2 = 1 - (exp(-a3* ((k-bass_note2) .^2)));
        guitar_rm_filter3 = 1 - (exp(-a3* ((k-bass_note3) .^2)));
        guitar_rm_filter4 = 1 - (exp(-a3* ((k-bass_note4) .^2)));
        %Apply filters in frequency domain
        mfygt = mfygt'.* guitar_rm_filter2;
        mfygt = mfygt.* guitar_rm_filter3;
        mfygt = mfygt.* guitar_rm_filter4;


        %filter by power
        %find 5 max power points after filtering out rhythm guitar
        [max_vals, idxs] = maxk(mfygt, 5);
```

```
268         %assume central frequency is lowest pitch of this set
269         central_freq = min(idxs);
270         cfvs(j) = k(central_freq);
271 
272         % Make Gaussian Filter centered at central frequency
273         a2 = .2;
274         filter = (exp(-a2* ((k-k(central_freq)) .^2)));
275         mfygt = mfygt.*filter; %apply Gaussian
276         fygt_spec(:,j) = fftshift(abs(mfygt));
277     end
278     G_full_song.(gspecs{i}) = fygt_spec; %save coefficients
279     G_full_song.(cent_freq_vals{i}) = cfvs;%save central frequencies
280 end
281 
282 
283 %% Plot
284 for i = 1:numel(gspecs) %make a different plot for each quarter
285     fygt_spec =G_full_song.(gspecs{i});
286     figure()
287     pcolor(tslide, ks, log(abs(fygt_spec)+1)), shading interp
288     set(gca, 'Ylim',[minfreq-10, maxfreq], 'Fontsize', 16)
289     colormap(hot)
290     xlabel('Time [sec]'); ylabel('Frequency [Hz]');
291     title_str = sprintf('Floyd Quarter %d Guitar Solo Spectogram', i);
292     title(title_str);
293     %colorbar
294 
295     %Label Relevant Notes
296 
297     fsl = yline(739.989,'-','F#5','LineWidth',.5);
298     fsl.LabelVerticalAlignment = 'middle';
299     fsl.Color = 'cyan';
300 
301     gl = yline(783.991,'-','G5','LineWidth',.5);
302     gl.LabelVerticalAlignment = 'middle';
303     gl.Color = 'cyan';
304 
305     dl = yline(622.254,'-','D#5','LineWidth',.5);
306     dl.LabelVerticalAlignment = 'middle';
307     dl.Color = 'cyan';
308 
309     el = yline(659.255,'-','E5','LineWidth',.5);
310     el.LabelVerticalAlignment = 'middle';
311     el.Color = 'cyan';
312 
313     fl = yline(698.456,'-','F5','LineWidth',.5);
314     fl.LabelVerticalAlignment = 'middle';
315     fl.Color = 'cyan';
316 
317     al = yline(880,'-','A5','LineWidth',.5);
318     al.LabelVerticalAlignment = 'middle';
319     al.Color = 'cyan';
320 
321     al = yline(440,'-','A4','LineWidth',.5);
```

```
322     al.LabelVerticalAlignment = 'middle';
323     al.Color = 'cyan';
324
325     dl = yline(587.33,'-','D5','LineWidth',.5);
326     dl.LabelVerticalAlignment = 'middle';
327     dl.Color = 'cyan';
328
329     cl = yline(523.251,'-','C5','LineWidth',.5);
330     cl.LabelVerticalAlignment = 'middle';
331     cl.Color = 'cyan';
332
333     cl = yline(554.365,'-','C#5','LineWidth',.5);
334     cl.LabelVerticalAlignment = 'middle';
335     cl.Color = 'cyan';
336
337     bl = yline(987.767,'-','B5','LineWidth',.5);
338     bl.LabelVerticalAlignment = 'middle';
339     bl.Color = 'cyan';
340
341     bl = yline(493.883,'-','B4','LineWidth',.5);
342     bl.LabelVerticalAlignment = 'middle';
343     bl.Color = 'cyan';
344 end
```

Appendix C Full Spectrograms

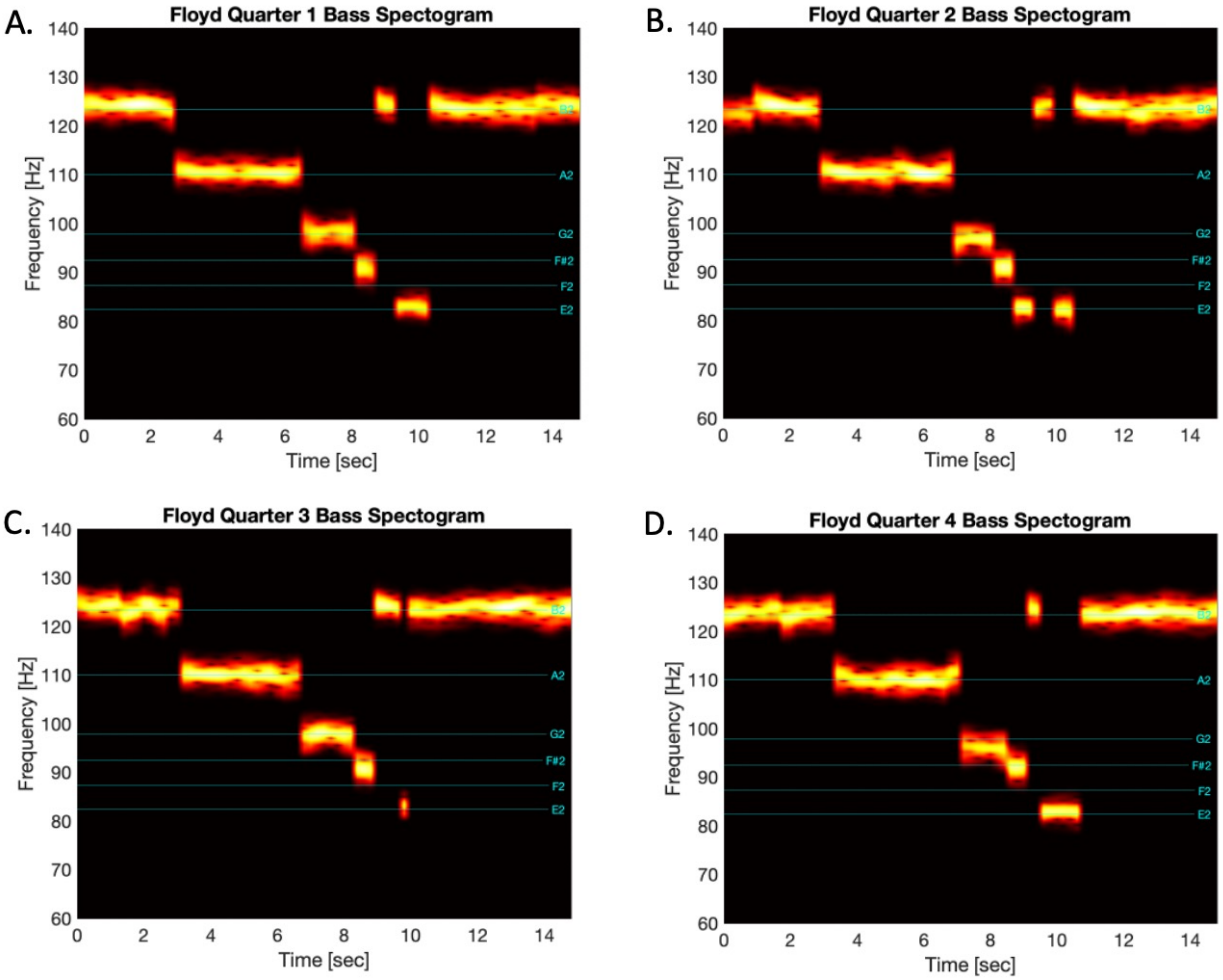

Figure 2: Spectrogram depicting notes played by the bass during each quarter of the "Comfortably Numb" excerpt. Relevant frequencies are labeled with their corresponding musical notes.

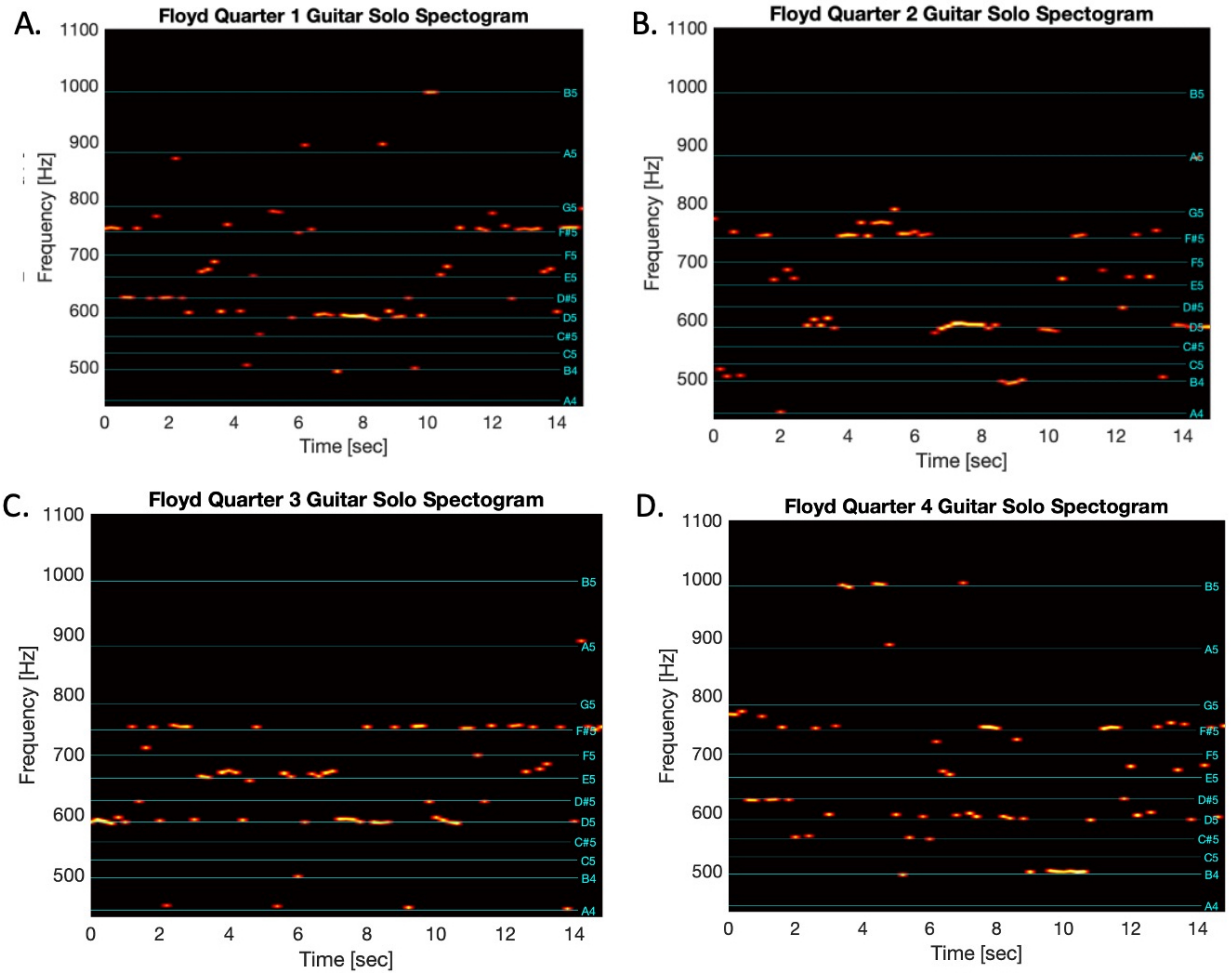

Figure 3: Spectrogram depicting notes played by the solo guitar during each quarter of the "Comfortably Numb" excerpt. Relevant frequencies are labeled with their corresponding musical notes.

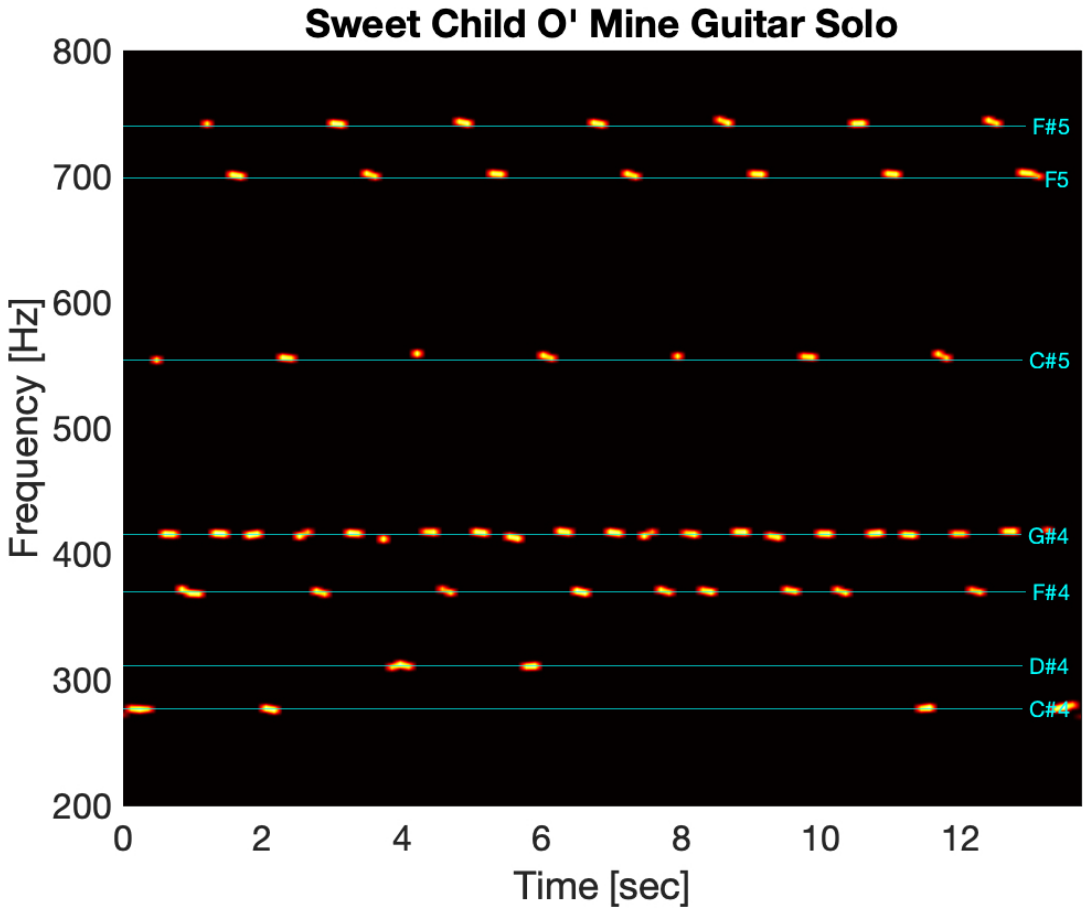

Figure 4: Spectrogram depicting notes played by the guitar in "Sweet Child O' Mine". Relevant frequencies are labeled with their corresponding musical notes.