Assignment 2: Spectrogram to Heaven

William Ojemann
AMATH 482
February 10th, 2021

Abstract

Digital signal processing has a wealth of potential uses, and the one with possibly the most room for creativity is the manipulation and analysis of music. This report outlines the implementation of a variety of time and frequency analysis methods to analyze samples of classic rock music. Filtering in the frequency domain and the application of a Gabor transform to the song clips are used to identify and isolate different instrumentals in the classic guitar solos from 'Sweet Child O' Mine' by Guns 'n Roses and 'Comfortably Numb' by Pink Floyd as well as transcribe their notes.

Introduction

While the Fourier transform forms the foundation for frequency analysis and digital signal processing, it has significant limitations. When analyzing a digital signal like a song, it is often important to not only visualize the frequency components that compose the signal but understand the frequency makeup of a signal at a certain time point. This is impossible with a simple Fourier transform because a signal loses all specificity in time when it is projected into the frequency space. Taking smaller chunks of the signal and performing a Fourier transform on those may seem like the obvious solution but taking smaller samples of the signal also means severely limiting the number of frequencies you can identify. To solve this problem and still follow the Heisenberg uncertainty principle, Dr. Dennis Gabor developed the Gabor transform. This mathematical process entails a traveling waveform that isolates small parts of the signal in time while not eliminating any samples, which maintains frequency resolution restricted to a narrower location in time.

The following computational background and methodology covers the use of the Gabor transform, as well as traditional frequency filtering, for the purpose of musical engineering. Using the guitar solos from 'Comfortably Numb' (Floyd) and 'Sweet Child O' Mine' (GNR) as data sets, Gabor transforms, and time-localized filtering methods will be applied to transcribe the solos as well as the baseline in Floyd and change the signal to isolate the baseline in Floyd. These processes, while fascinating and commonplace in sound production technology, require a background in signal processing to fully understand, which will be provided.

Theoretical Background

All equations are adapted from lecture notes. The Gabor transform, or short-time Fourier transform (STFT) is the basis for the majority of the analysis performed in this report. The traditional continuous Fourier transform (1) is a projection of a signal from the time or space dimension into the frequency space, and its inverse (2) does the opposite projecting a signal from its frequency components back into the time or space domain.

$$\hat{f}(k) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} f(x)e^{-ikx}dx \qquad \text{eq. 1}$$

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

Because a Fourier transform sums across infinite time and its inverse across infinite frequency, the information produced by the transform has no spatial specificity. The frequencies shown in the projection could be occurring at any point in time in the signal. While this presents a problem that may seem solvable in the world of digital data and the discrete Fourier transform (DFT) (3), further problems arise.

$$X_k = \sum_{n=0}^{N-1} x_n e^{\frac{2\pi ikn}{N}} \quad \text{eq. 3}$$

The DFT can be applied over a limited spatial range of, but in limiting the coverage in time to N data points the projection also limits the coverage in the frequency space to N frequencies. While still obeying the Heisenberg uncertainty principle, the STFT (4) allows for comprehensive frequency analysis limited to a window in time.

$$\widetilde{f_g}(\tau, k) = \int_{-\infty}^{\infty} f(t)g(t-\tau)e^{-ikt}dt \quad \text{eq. 4}$$

The short-time aspect of the STFT is encompassed in the $f(t)g(t-\tau)$ portion of the equation where a windowing function g(t) is centered around a time point τ and multiplied by the signal (**figure 1**). This reduces everywhere outside the window of interest to zero and isolating that portion of the signal while still maintaining the same frequency resolution. The STFT is still limited by the uncertainty principle in the fact that the information about the frequencies of a signal will be less comprehensive as the window size decreases. Conversely, as the size of the window approaches infinite, the STFT simply becomes a standard Fourier Transform.

While STFT and Gabor transform have been used interchangeably, a Gabor transform is technically a STFT with a Gaussian window (5) used as the window function. For the purpose of this report, the two terms are interchangeable and any STFT will be performed using a Gaussian window.

$$G(k) = e^{-a(t-\tau)^2} \quad \text{eq. 5}$$

The STFT is an incredibly useful tool for signal analysis, but practical application requires a discretized version of the transform.

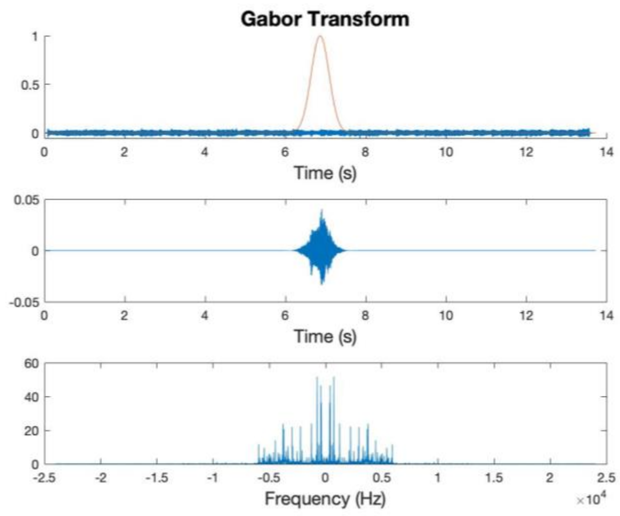


Figure 1: *(top) Gaussian window and original signal - GNR. (middle) Windowed signal. (bottom) FFT of the windowed signal, showing frequencies only for the small portion of time.*

As a result, the discrete STFT (6) was developed to address the needs of applied mathematicians and physicists with the discrete variables $k = m\omega_0$ for frequency and $\tau = nt_0$ for time.

$$\widetilde{f_g}(m, n) = \int_{-\infty}^{\infty} f(t)g(t-nt_0)e^{2\pi i m\omega_0 t}dt \quad \text{eq. 6}$$

When the discrete STFT is applied to a signal, it produces a set of frequency components for each τ, and this three dimensional can most easily be visualized using a spectrogram with color corresponding to the energy of the frequencies at different values of τ. For the signal analysis performed in this report, because the signal of interest is music with notes measured in Hertz, the frequencies are scaled from radians to Hertz by dividing the frequencies $2\pi m\omega_0$ by 2π. The STFT can also be inverted similar to the DFT, but the inverse transform is not utilized in this assignment as all analysis is performed in the frequency domain. Because the assignment outlines work in the frequency domain, it is important to note a phenomenon called the overtones or harmonics of a frequency. When a note is played, integer multiples of its frequency can also be heard and appear in the frequency decomposition of that sound. Because of this, the transcription and analysis of music requires filtering out overtones in the frequency space.

In addition to the STFT, this paper outlines the use of a Shannon window in frequency space, which is the combination of two translated Heaviside functions (7) to create what is effectively a square wave.

$$H(t) = \begin{cases} 0, & t < 0 \\ 1, & t \geq 0 \end{cases} \qquad \text{eq. 7}$$

Algorithm Implementation and Development

Meeting the goals of this assignment, finding the notes for the guitar solo in GNR, finding the notes for the bass line in Floyd, isolating the bass line in Floyd, and finding the notes for the guitar solo in Floyd, requires extensive use of signal processing tools applied in MATLAB. To analyze the guitar solo in GNR, the .m4a sound file was imported using MATLAB's 'audioread' function (**figure 2**), extracting the waveform and the sampling frequency. Applying frequency analysis starts with formatting the data using the signal and its sampling frequency. The length of the signal in time, the number of samples, the time space of the signal, and a grid containing the frequency space of the signal were all extracted with the internal 'formatting' function. The frequency space of the signal was transformed from radians/sec to Hz by dividing the frequency space by 2π. To visualize the STFT process, the Gabor transform was applied to the signal with a tau set of 101 unique tau's and a Gaussian scale value of a = 10 (5) using the internal 'gabor_process' function.

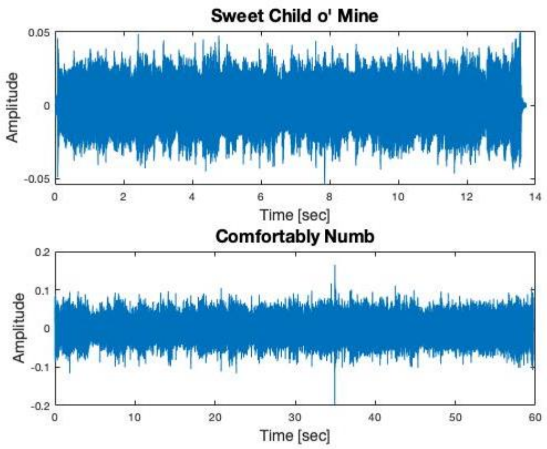


Figure 2: *Wave forms for both the samples of Sweet Child o' Mine and Comfortably Numb.*

The actual implementation of the STFT to reveal the frequency components of the GNR sample iterated through the same tau set and multiplied the signal by a Gaussian window with scaling a = 10 centered at each tau.

The windowed signal was then projected into the frequency space using an FFT and the absolute value of the frequencies – to maintain any complex information – were shifted out of MATLAB's indexing. the resulting frequencies were then filtered using logical indexing to include frequencies from 0 Hz to 790 Hz, retaining the guitar solo but filtering out any overtones. The STFT application and subsequent spectrogram plotting were implemented using the internal 'spec_plot' function. After fine-tuning the STFT hyper parameters (**Appendix B**), the notes of the guitar solo could be extracted from the resulting spectrogram by finding the maximum energy frequency for each value of tau and matching it to a note-frequency comparison chart. Because the frequencies were returned in index values, they had to be converted to notes using the GNR frequency grid.

The initial extraction of the bass line from Floyd followed a very similar process as the GNR guitar solo with the exception of the

STFT hyperparameters and frequency filtering. The scaling factor of the Gaussian window was set to 15 and the window slid over 251 values of tau. To isolate the bass line from the sample of music and avoid overtones, the frequencies were filtered with logical indexing to only include frequencies in the range of 50 Hz to 250 Hz. Because the musical sample from Floyd was much longer than that of GNR, the sample was split up to eight sections and the same analysis applied to each. Even after fine tuning the hyper parameters of the windowing function, there was still a large amount of noise and overtones from lower frequencies in the bass line present within the bass window. Because the specific notes could not be identified with a high level of qualitative certainty, further isolation was required.

To better identify the primary frequency in each spectrogram, after applying the Gabor transform, taking the FFT of the data, and filtering it to the appropriate range, the frequency spectrum was multiplied by a Shannon filter. The filter was created with logical indexing that provided a window of ones around the maximum energy frequency for each spectrogram where the width was specified by a manually tuned parameter. The additional filtering process was implemented using the internal 'isolation' function. With the newly filtered signal, the notes for the bass line of Floyd could be transcribed by similarly finding the maximum energy frequency for each tau and projecting the signal onto the Floyd frequency grid.

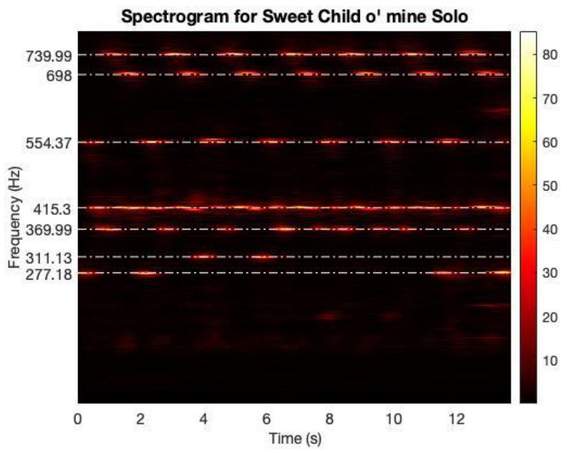


Figure 3: *Spectrogram showing the guitar solo on Sweet Child o' Mine acquired using a STFT.*

To achieve the last goal, which was to transcribe the Floyd guitar solo, the same isolation and filtering process was used as on the bass line but in a higher frequency range from 250 Hz to 1000 Hz. To more easily analyze the fast guitar solo, the song sample was again split up but this time into 10 different sections of equal length. The same process of STFT application, maximum frequency identification, filtering, and projection into the Floyd frequency grid was then followed for each sample. With the score for each section, the notes for the guitar solo on Floyd were partially identified by matching frequencies on the provided note-frequency conversion chart.

Computational Results

The result of applying a STFT to the GNR song sample and filtering the resulting spectrograms to an appropriate frequency spectrum is the identification of the notes in the legendary guitar solo. While the spectrogram allows for the visual identification of the correct score (**figure 3**),

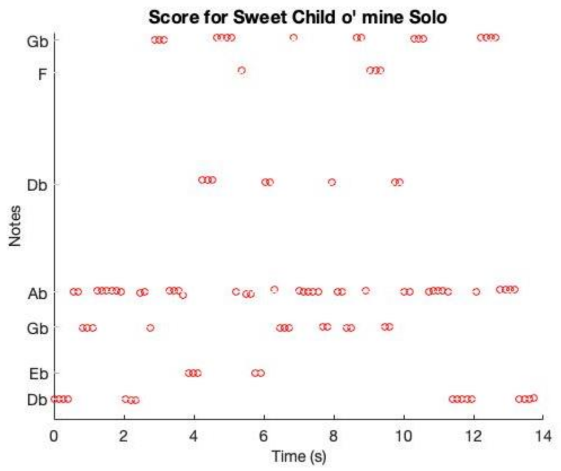


Figure 4: *Notes taken from the maximum frequency at each time point in the spectrogram representing the guitar solo on Sweet Child o' Mine.*

using the maximum energy frequency introduces some errors in higher frequency notes early in the song (**figure 4**). The notes in the guitar solo also identify the musical key of the song to be Db.

Applying a STFT to the sample Floyd song and filtering the resulting spectrograms to an appropriate lower frequency range results in the song's baseline being obscured in part by noise and overtones (**figure 5**). After applying a filter to the spectrograms around the maximum frequency, the Comfortably Numb bass line can be more easily identified from the spectrogram (**figure 6**) and the score of the bass line can be reconstructed with some potential error from low frequency overtones and noise (**figure 7**).

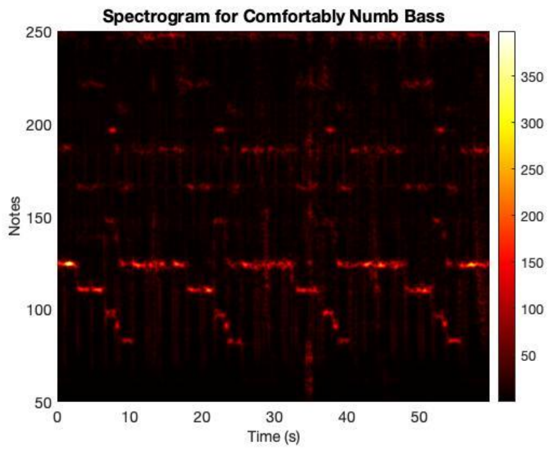
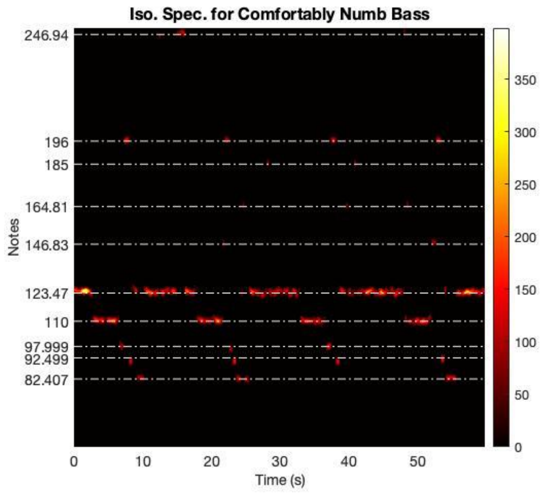


Figure 5: *Spectrogram for the bass line in Comfortably Numb acquired using a STFT.*

Because the guitar solo is a fast solo, the resulting spectrograms and score plots are split up into ten different sections for readability (**Appendix C**). By applying an STFT to the Floyd song sample, filtering the frequencies into the appropriate frequency for the guitar and further filtering around each maximum frequency allows for the rough identification of the beginning of the guitar solo from the spectrogram (**figure 8**). Using these notes, the approximate score of the solo can be plotted (**figure 9**). Because of the potential presence of prominent overtones

Figure 6: *Spectrogram for the bass line in Comfortably Numb acquired using a STFT and further filtered with a band-pass Shannon filter.*



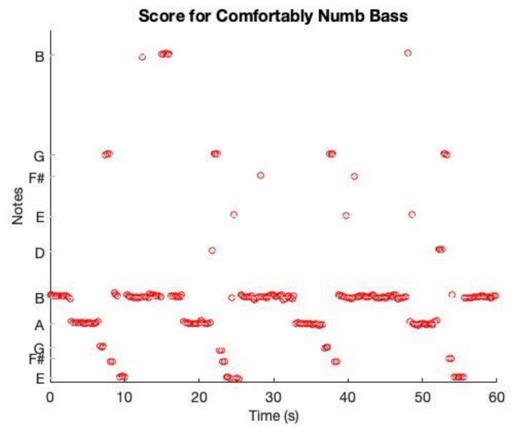
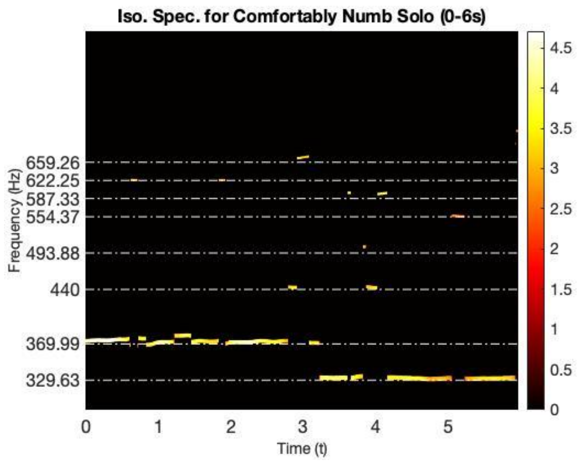


Figure 7: *Notes for the bass line in Comfortably Numb.*

and chords being played during the solo the resulting notes for the first part of the solo has low qualitative confidence. Based on the score, the guitar solo is likely in the key of B minor.

Figure 8: *Spectrogram for the guitar solo in Comfortably Numb acquired using a STFT and further filtered with a band-pass Shannon filter.*



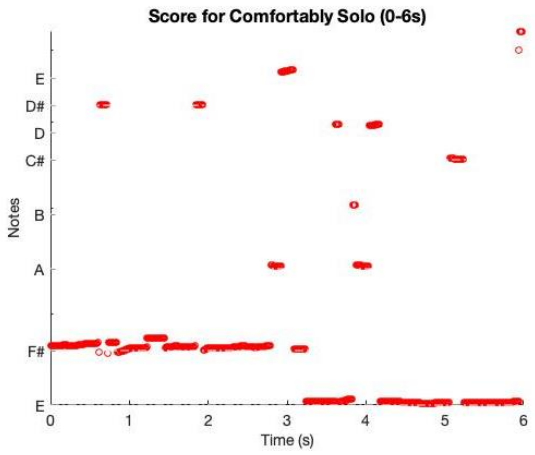


Figure 9: *Notes for the first portion of the guitar solo in Comfortably numb.*

Summary and Conclusions

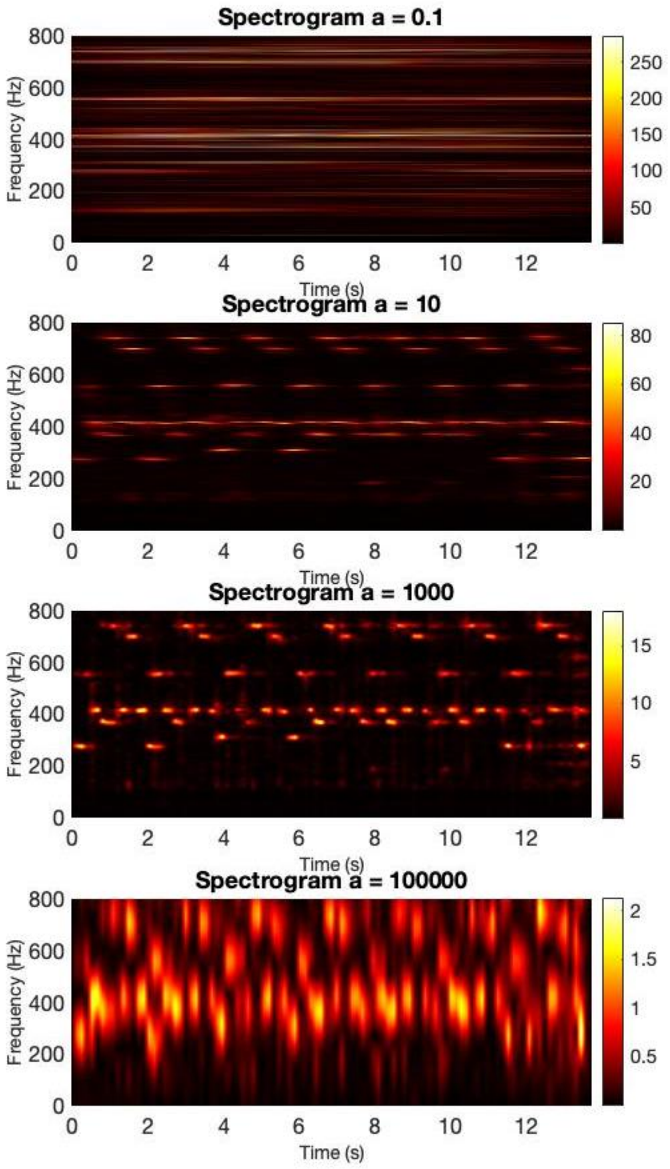
This project again demonstrates one of the many real-world applications of signal processing. The use of spatially sensitive frequency analysis tools such as the Gabor transform, spectrogram generation, and frequency filtering allow for both the analysis and manipulation of sound waves. This paper demonstrates the application of powerful signal processing methods – and their associated challenges – towards identifying notes of famous guitar solos from 'Comfortably Numb' and 'Sweet Child O' Mine' and isolating the underlying baseline of the Pink Floyd song. These tools have a myriad of uses within the field of sound engineering and the work outlined in this report is a small subsect of the creative possibility frequency analysis offers.

Appendix A – MATLAB Functions

[x, Fsx] = audioread('filename.m4a') – loads an audio file returning the file as an array, x, and its sampling frequency, Fsx, used to load the given song samples.
linspace(start, end, int) – creates an array of variables between the start and end value with int data points, used to create the space and frequency dimensions with the number of Fourier modes taken by the data acquisition unit.
fftshift(K) – shifts the zero-frequency component of a frequency spectrum to the center of the array, used to shift the zero-frequency component to the center of our frequency dimensions after initializing.
abs(x) – takes the absolute value of x, or if x is an array then it takes the absolute value of each element in x, used to consider the complex component of a data point at many points throughout the implementation.
fft(t) – takes the fast Fourier transform of the array t, used to calculate the FFT of the Gabor windowed signal.
[~, maxIndex] = max(Y, [], 'all', 'linear') – finds the maximum element in each column of Y and returns the the index of that maximum value, used to find the frequency of the note being played at that value of tau after applying a Gabor filter.
plot(t,y) – plots the signal y against its time domain t, used to plot the songs before processing and analysis.
scatter(t,k) – plots the individual data points as dots in the frequencies k against its time domain t, used to plot the notes (frequencies) in a score.
pcolor(x,y,Z) – plots a heatmap of the matrix Z on the axes x and y with the value of each element in the matrix corresponding to the color portrayed on the heatmap, used to create spectrograms for the results of the STFT.
formatting(y, FS) – Internal function used to extract relevant features for signal processing from the signal and its sampling frequency.
gabor_process(t,ks,y,tauset,a,fignum) – Internal function used to perform and plot the Gabor transform being applied to a signal.
spec_plot(t,ks,y,tauset,a,fignum,ylimits) – Internal function used to plot spectrograms and apply band pass filters to limit the presence of overtones in a spectrogram.
score_plot(t,notes,notelabels) – Internal function used to add specific frequencies and notes to plot.
Isolation(ks,spec,score,tauset,filter_width,fignum) – Internal function used to take a spectrogram and isolate the maximum frequency at each sample and return a filtered spectrogram with the isolated notes.

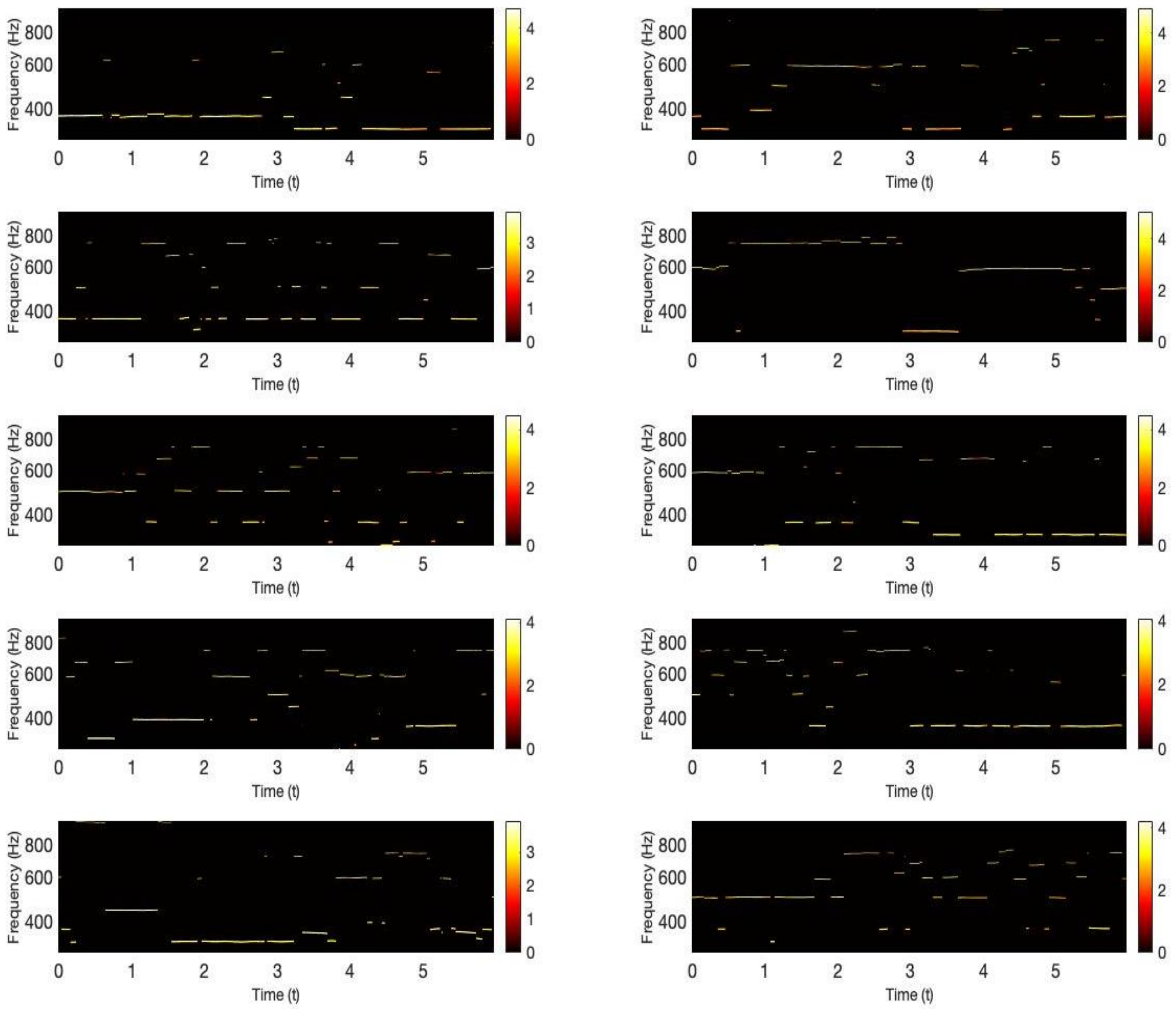
Appendix B – Hyper Parameter Tuning for STFT

Plots of tuning hyper parameters in the STFT. This plot modulates a, or the scaling factor for the width of the Gaussian window, which decreases with larger a's. These plots demonstrate the tradeoff between frequency and time resolution with smaller a's having defined frequencies and larger a's having defined time. The tuning process finds a balance.



Appendix C – Band-Pass Filtered Spectrograms for Floyd Guitar Solo

Showing 6 second intervals of the solo increasing from left to right, top to bottom.



Appendix D – MATLAB Code

```
% Assignment 2
% William Ojemann
% AMATH 482
% 1/31/2021
clear; close all; clc;
%%
% Audio Clip Plotting
[gnr, Fsgnr] = audioread('GNR.m4a');
[floyd, Fsfloyd] = audioread('Floyd.m4a');
figure(1)
set(gca,'Fontsize',14)
subplot(2,1,1)
plot((1:length(gnr))/Fsgnr,gnr);
xlabel('Time [sec]','Fontsize',14); ylabel('Amplitude','Fontsize',14);
title("Sweet Child o' Mine",'Fontsize',16);
subplot(2,1,2)
plot((1:length(floyd))/Fsfloyd,floyd);
xlabel('Time [sec]','Fontsize',14); ylabel('Amplitude','Fontsize',14);
title('Comfortably Numb','FontSize',16);
%% Guns 'n Roses
clc
[gnr, Fsgnr] = audioread('GNR.m4a');
p8 = audioplayer(gnr,Fsgnr);
playblocking(p8);

[t, ks, ~, ~, L] = formatting(gnr, Fsgnr);
a = 10;
n_taus = 101;
tauset = 0:L/(n_taus-1):L;
fignum = 2;
gabor_process(t,ks,gnr,tauset,a,fignum)

fignum = 3;
ylimits = [0 800];
[Y_gnr,filter_indices] = spec_plot(t,ks,gnr,tauset,a,fignum, ylimits);
set(gca,'Fontsize',14)
title("Spectrogram for Sweet Child o' mine Solo",'Fontsize', 16)
ylabel('Frequency (Hz)','Fontsize', 14)
xlabel('Time (s)','Fontsize', 14)
[~,score] = max(Y_gnr,[],1);
ksf = ks(filter_indices);
scoregnr = ksf(score);
score_plot(t,sort([277.18,554.37,415.3,369.99,739.99,698,311.13]),{});
figure(100)
hold on
scatter(tauset,scoregnr,'r')
set(gca,'Fontsize',14)
ylabel('Notes','Fontsize', 14)
xlabel('Time (s)','Fontsize', 14)
title("Score for Sweet Child o' mine Solo",'Fontsize', 16)
score_plot(t,sort([277.18,554.37,415.3,369.99,739.99,698,311.13]),{'Db','Eb',
'Gb','Ab','Db','F','Gb'});
%% Hyper Parameter Tuning
[gnr, Fsgnr] = audioread('GNR.m4a');
[t, ks, ~, ~, L] = formatting(gnr, Fsgnr);
```

```
a = [.1, 10, 1000, 100000];
n_taus = 101;
tauset = 0:L/(n_taus-1):L;
fignum = 106;
ylimits = [0 800];
for j = 1:length(a)
    subplot(4,1,j)
    spec_plot(t,ks,gnr,tauset,a(j),fignum, ylimits);
    xlabel('Time (s)', 'Fontsize', 14)
    ylabel('Frequency (Hz)', 'Fontsize', 14)
    title(['Spectrogram a = ' num2str(a(j))]);
end
%% Pink Floyd
% %clear; close all; clc;
[floyd, Fsfloyd] = audioread('Floyd.m4a');
p8 = audioplayer(floyd,Fsfloyd);
playblocking(p8);
%% Baseline Isolation
clc
[floyd, Fsfloyd] = audioread('Floyd.m4a');
a = 15;
n_taus = 251;
[t, ks, floyd, n, L] = formatting(floyd, Fsfloyd);
ylimits = [50 250];
fignum = 4;
tauset = 0:L/(n_taus-1):L;
[Y_floyd,filter_indices] = spec_plot(t,ks,floyd,tauset,a,fignum, ylimits);
set(gca,'Fontsize',14)
ylabel('Notes','Fontsize', 14)
xlabel('Time (s)','Fontsize', 14)
title("Spectrogram for Comfortably Numb Bass",'Fontsize', 16)
[~,score] = max(Y_floyd,[],1);
ksf = ks(filter_indices);
scorefloyd = ksf(score);
filter_width = 4;
hold on
[filter_spec] = isolation(ksf,Y_floyd,scorefloyd,tauset,filter_width,5);
score_plot(t,[82.407,92.499,97.999,110,123.47,146.83,164.81,185,196,246.94],{
});
set(gca,'Fontsize',14)
ylabel('Notes','Fontsize', 14)
xlabel('Time (s)','Fontsize', 14)
title("Iso. Spec. for Comfortably Numb Bass",'Fontsize', 16)

[~,filter_score] = max(filter_spec,[],1);
filter_score = ksf(filter_score);
figure(101)
hold on
scatter(tauset,filter_score,'r')
score_plot(t,[82.407,92.499,97.999,110,123.47,146.83,164.81,185,196,246.94]..
.
    ,{'E','F#','G','A','B','D','E','F#','G','B'});
set(gca,'Fontsize',14)
ylabel('Notes','Fontsize', 14)
xlabel('Time (s)','Fontsize', 14)
title("Score for Comfortably Numb Bass",'Fontsize', 16)
%% Guitar Solo Identification
```

```
clc
[floyd, Fsfloyd] = audioread('Floyd.m4a');
l = length(floyd);
a = 100;
ylimits = [300 1000];
n_taus = 701;
[~, ~, floyd, ~, ~] = formatting(floyd, Fsfloyd);
figs = 10;
score_set = nan(figs,n_taus);
figure(7)
%subplot(figs/2,2,1)
[t, ks, floyds, ~, L] = formatting(floyd(1:floor(l/figs)), Fsfloyd);
fignum = 7;
tauset = 0:L/(n_taus-1):L;
[Y_floyd,filter_indices] = spec_plot(t,ks,floyds,tauset,a,[], ylimits);
[~,score] = max(Y_floyd,[],1);
ksf = ks(filter_indices);
score_set(1,:) = ksf(score);
filter_width = 5;
isolation(ksf,Y_floyd,score_set(1,:),tauset,filter_width,fignum);
score_plot(t,[329.63,369.99,440,493.88,554.37,587.33,622.25,659.26],{});
set(gca,'Fontsize',16, 'YScale', 'log')
ylabel('Frequency (Hz)','Fontsize', 14)
xlabel('Time (t)','Fontsize', 14)
title("Iso. Spec. for Comfortably Numb Solo (0-6s)",'Fontsize', 16)

figure(104)
hold on
set(gca,'Fontsize',16, 'YScale', 'log')
scatter(tauset,score_set(1,:),'r')
score_plot(t,[329.63,369.99,440,493.88,554.37,587.33,622.25,659.26],...
    {'E','F#','A','B','C#','D','D#','E'});
set(gca,'Fontsize',14)
ylabel('Notes','Fontsize', 14)
xlabel('Time (s)','Fontsize', 14)
title("Score for Comfortably Solo (0-6s)",'Fontsize', 16)
for j = 2:figs
    %fignum = 6+j;
    subplot(figs/2,2,j)
    [t, ks, floyds,~,~] = formatting(...
        floyd(floor(l/figs*(j-1)):floor(l/figs*j)-1), Fsfloyd);
    [Y_floyd,filter_indices] = spec_plot(t,ks,floyds,tauset,a,fignum,
ylimits);
    [~,score] = max(Y_floyd,[],1);
    ksf = ks(filter_indices);
    score_set(j,:) = ksf(score);
    isolation(ksf,Y_floyd,score_set(j,:),tauset,filter_width,fignum);
    set(gca,'Fontsize',16, 'YScale', 'log')
    ylabel('Frequency (Hz)','Fontsize', 14)
    xlabel('Time (t)','Fontsize', 14)
    score_plot(t,5*[82.407,92.499,97.999,110,123.47,164.81,185,196],{})
end
%% Plotting Guitar Solo Notes
ltauset = 0:l/Fs/(n_taus*figs-1):l/Fs;
linscore = [];
for j =  1:figs
    linscore = [linscore score_set(j,:)];
```

```
end

figure(102)
scatter(ltauset,linscore)
set(gca,'Fontsize',16, 'YScale', 'log')
ylabel('Frequency (Hz)','Fontsize', 14)
xlabel('Time (t)','Fontsize', 14)
Title('Comfortably Numb Guitar Solo')
%% Functions
% Initial Data Formatting -- internal function
function [t, ks, y, n, L] = formatting(y, Fs)
    if mod(length(y),2) == 1
        y = y(1:end-1);
    end
    % Feature Extraction
    L = length(y)/Fs; % record time in seconds
    n = length(y);
    t2 = linspace(0,L,n+1); t = t2(1:n);
    k = (1/L)*[0:(n)/2-1 -(n)/2:-1]; ks = fftshift(k);
end

% Plotting Filtering Process -- internal function
function gabor_process(t,ks,y,tauset,a,fignum)
    if ~isempty(fignum)
        figure(fignum);
    end
    for tau = tauset
        giter = exp(-a*(t-tau).^2);
        subplot(3,1,1)
        hold on
        plot(t,y)
        plot(t,giter);
        xlabel('Time (s)', 'FontSize', 14)
        title('Gabor Transform', 'FontSize', 16);
        yf = giter'.*y;
        subplot(3,1,2)
        plot(t,yf)
        yft = fft(yf);
        xlabel('Time (s)', 'FontSize', 14)
        subplot(3,1,3)
        plot(ks,fftshift(abs(yft)))
        xlabel('Frequency (Hz)', 'FontSize', 14)
        drawnow
        pause(0.1)
        clf
    end
end

% Plotting Spectrogram -- internal function
function [y_spec,filter_indices] = spec_plot(t,ks,y,tauset,a,fignum, ylimits)
    filter_indices = (ks >= ylimits(1) & ks <= ylimits(2));
    y_spec = nan(sum(filter_indices),length(tauset));
    for j = 1:length(tauset)
        giter = exp(-a*(t-tauset(j)).^2);
        yf = y.*giter';
        yft = fft(yf);
        yft = fftshift(abs(yft));
        yf = yft(filter_indices);
        y_spec(:,j) = yf;
    end
    if fignum
        figure(fignum)
        pcolor(tauset,ks(filter_indices),y_spec);
        set(gca,'ylim',ylimits,'Fontsize',16)
        shading interp
        colormap(hot)
        colorbar
    end
end

% Note Label Plotting -- internal function
function music = score_plot(t,notes,notelabels)
    hold on
    music = nan(length(notes),2);
    tset = [t(1) t(end)];
    for note = notes
       plot(tset, [note note], 'w-.','LineWidth',1);
    end
    yticks(notes);
    if ~isempty(notelabels)
        yticklabels(notelabels);
    end
    hold off
end

% Musical Part Isolation -- internal function
function [filter_spec] = isolation(ks,spec,score,tauset,filter_width,fignum)
    filter_spec = nan(length(ks),length(tauset));
    for j = 1:length(tauset)
        ygt = spec(:,j);
        note = score(j);
        filter_indices = ks < (note+filter_width/2) & ks > (note-
filter_width/2);
        ygft = ygt'.*filter_indices;
        filter_spec(:,j) = ygft;
    end
    figure(fignum)
    pcolor(tauset,ks,log(filter_spec+1));
    set(gca,'Fontsize',16, 'YScale', 'log')
    shading interp
    colormap(hot)
    colorbar
end
```