Using the Gabor Transform to Determine Musical Scores

Ben Francis

Febryary 10, 2021

Abstract

The audio signals from two songs, "Sweet Child O' Mine" and "Comfortably Numb", are converted to their musical scores by use of the Gabor Transform and the construction of a spectrogram. In "Comfortably Numb", the bass and guitar scores are constructed separately. While promising, these techniques also require the use of human intervention for higher confidence results.

1 Introduction and Overview

Music sits directly at the interface of the time and frequency domains. The perceptual experience is almost entirely determined by the frequency of the sensory signal, yet it is entered into the auditory system as a time signal. As such, there must be some sort of spectral method undertaken by the auditory sensory system.

In trying to recreate this, the previously introduced theory of Fourier transforms is immediately considered. An oscillatory kernel can extract the frequency components from a time signal. However, there is a substantial drawback in this transform. The Fourier transform is generally applied to an entire signal, recorded over some duration. This transform can then determine the degree to which high and low frequencies contribute to the signal. However, it cannot determine where the different frequency components exist within the signal.

If the signal under consideration is a song, then while it can determine which notes are played throughout the song, it cannot determine the ordering of these notes. The musical experience is characterized not just by the tones present in the piece, but also by their time-dependent composition, resulting in melody and other important musical forms. To resolve this problem, consider further the human auditory experience. While the auditory experience is in the frequency domain, it changes rapidly with time. It is much like the auditory system is continuously sampling new Fourier transforms from the continuously changing pressure signals. This is the basic idea of the Gabor transform.

The intuitive idea behind the Gabor transform is to divide the signal into smaller time slices and then take the Fourier transform of each time slice. This gives a dynamic frequency domain representation of the signal, providing a simple computational analog of the biological auditory process.

2 Theoretical Background

2.1 The Gabor Transform

The Fourier transform (FT) takes a signal, represented as a function $f(t)$, multiplies it by an oscillatory kernel, and integrates across the time domain. As such, all information about the time domain is lost, and so the frequency components given by the transform cannot be placed in different times. This is akin to knowing there were ice ages in the geological past, but not being able to say at all when they were.

The Gabor transform mitigates this problem by applying an additional time-dependent kernel to the transform, thereby making the output a function of both frequency and time. Instead of a one-dimensional frequency space output, this results in a two-dimensional frequency-time output. This results in a two-dimensional graph called a spectrogram. The goal of the kernel is to selectively filter unique slices of time,

such that the oscillatory kernel acts on each unique time slice to determine the oscillatory behavior at that point in time. When this is done for all points in time, the manner in which the frequencies change over time can be understood, analogous to the human auditory system.

The method by which the Gabor transform performs this time dynamic process is by using a kernel that can perform time-sliding. This kernel has the form $g(t-\tau)$. If g is a function with a center at zero, such as a Gaussian, this has the effect of shifting the center as τ, or t depending upon convention, increases.

While there are many different kernels that can be used, the Gaussian kernel will be examined here. There are benefits and downsides to different filters. For instance, the Shannon filter uses double-sided step function to make everything outside one window of the signal equal to zero. While having the benefit of not affecting the signal at that time window, it results in sharp corners which cause Gibbs phenomena upon taking the Fourier transform. One benefit of the Gaussian transform is that while it has a center which can be time-slided, it gradually diminishes in amplitude amongst neighboring time points, still receiving some influence from their behavior.

The equation for the Gabor transform of a signal $x(t)$ is then: [3]

$$G(k,t) = \int_{-\infty}^{\infty} f(\tau) g(t-\tau) e^{-ik\tau} d\tau, \tag{1}$$

The discrete Gaussian kernel will be examined here, so that the $g(t-\tau)$ term is rewritten as: [3]

$$g(t-b) = e^{-a(t-b)^2} \tag{2}$$

a is the width parameter. Here a is inversely proportional to the width or variance of the Gaussian. This width determines the temporal versus frequency information of the spectrogram. If the width parameter is smaller, then there will be lower temporal resolution and higher frequency resolution. The spectrogram will more closely resemble the Fourier transform. If the width parameter is larger, there will be higher temporal resolution and less frequency resolution. The spectrogram will more closely resemble the signal. There is a minimal trade-off between the two called the Gabor limit, which is a generalization of the Heisenberg uncertainty principle from quantum mechanics [1]. Usually a midpoint between these two extremes is most beneficial, with fine-tuning necessary for the particular problem at hand. Gaussians have unique properties related to being intermediate between the time and frequency domains. The discrete Gabor transform sums over a discrete set of b instead of integrating across a continuum of τ. The distance between each b results in a non-trivial effect, and so b is known as the translation parameter. If b is small, then each successive Gaussian will overlap the previous Gaussian as the window slides across time, resulting in oversampling. This results in an overall higher resolution and better spectrogram, though it is less efficient since more time steps are needed. Conversely, a larger translation parameter results in a lower resolution spectrogram, but is more efficient. These two parameters are the primary tunable aspects of the Gabor transform, in addition to the choice of kernel.

2.2 Musical Analysis

The goal of this paper is to extract musical notes from sound signals from two songs. As such, the structure of music will play an important role. Western music is composed of 12 notes: C, C#, D, D#, E, F, F#, G, G#, A, A#, and B. These notes are primarily chosen due to their characteristic sound to the human mind, though each also has the property that its frequency is $2^{1/12}$ larger than the previous note. They are periodic in the sense that after B is another C, except now C is one octave higher, and by definition has a frequency twice as great, since $2^{1/12^{12}}C = 2C$. To distinguish these two C notes, each is given a number, such that $C3$ is one octave, which is twelve notes, higher than $C2$. A grand piano typically has a note range from $A0$ to $C8$, corresponding to the frequencies 27.5 Hz and 4186.01 Hz, respectively. Songs also typically have a periodicity in melody, such that notes are played and held over similar time scales throughout the song, resulting in, or resultant from, a beat. Knowing the beat of a song, usually expressed in beats-per-minute (bpm) is helpful in identifying the notes in a song. While each note has an associated frequency, when a note is struck on a real instrument it also results in frequencies of higher octaves, or power-of-two times larger

frequencies, though with less amplitude. This adds substantial noise to the frequency domain, especially when the original note is not as pronounced, as will be encountered when trying to determine the guitar score in *Comfortably Numb*. When the spectrogram is created, one heuristic to look at for which note is actually played is to examine which frequency has the highest amplitude. However, since the signals and their transforms are noisy, due to additional instruments and unique timbre and other musical factors of the instruments, it is important to note this is only a heuristic.

There are four tasks of interest: (1) Determine the notes for the guitar intro to *Sweet Child O' Mine* by Guns and Roses, (2) Determine the notes for the bass in the solo of *Comfortably Number* by Pink Floyd, and (3) Determine as many notes as possible for the guitar solo in *Comfortably Numb*. For all of these tasks, it is important to note the plausible range of guitars and bass guitars, as well as the experimental range within each song. The highest note on a standard bass guitar is D#4 with a frequency of 311.13 Hz. The lowest note on a guitar is E3, with a frequency of 164.81 Hz. [2] However, in practice bass songs rarely use their notes above E3, so 164.81 Hz is a good heuristic cutoff between the two instruments, where any exceptions can be noted by seeing the melody of the bass move out of this range. In the clips analyzed here, this heuristic holds true. The playing style of instrumentalists is also a factor for transcribing the sound waves into musical score. The *Sweet Child O' Mine* intro is clean and crisp, while the *Comfortably Numb* bass is slow and repetitive, making both relatively easy to read off. In contrast, the guitar solo in *Comfortably Numb* is fast and performed by David Gilmore, who is known to use many slides and bends, as well as other difficult to analyze techniques. As such, it makes for a more difficult problem and increases the risk of missed notes.

3 Algorithm Implementation and Development

Algorithm 1: Create Spectrogram and Determine Maximum Frequencies

Data: Audio file
Result: Spectrogram and list of maximum frequencies at each Gabor Transform translation interval
`Signal, Sampling rate` ← `music.m4a` ;
`n` ← Number of data points;
`L` ← Length of recording;
Define time domain as `n` points over `L` seconds;
Define frequency domain, only looking at positive values;
Define notes and their corresponding frequencies;
Define plausible frequencies for this problem depending on if bass or guitar;
`a` ← Gaussian width;
`b` ← Translation parameter;
Define grid `t_slide` for applying filter by dividing `L` into steps of size `b`;
for j = *1:t_slide* **do**
    Define Gaussian filter with width `a`, centered at `t_slide`(j);
    `Signal` ← Multiply signal by filter;
    Signal ← Take FFT of signal;
    Signal ← Shift signal and take absolute value;
    Signal ← Remove negative frequencies, and filter only plausible frequencies;
    Append`Signal` to `Spectrogram` ;
    Determine index of frequency with maximum amplitude;
    Add frequency corresponding to this index to maximum frequency list;
Plot spectrogram;
Plot maximum frequencies;
Overlay all frequencies and notes on graphs to reference notes;

Algorithm 2: Map frequencies to note names

```
Data: List of maximum frequencies in spectrogram
Result: List of note names
proximity_threshold ← 2;
Define list of note names from C1 to C#6;
Define all frequencies corresponding to note names;
for All frequencies we want to convert to notes do
    for All frequencies corresponding to notes do
        if Frequency is within proximity_threshold of frequency corresponding to a note then
            if This is the first note or the note isn't a repeat then
                Add corresponding note to list of notes;
```

3.1 Algorithm Development

3.1.1 Setup

While there are three different problems of interest, each of the problems shares the same core, which is given in Algorithm 1. The primary difference rests in determining what the plausible frequency range is for each song, and in the case of the guitar solo, applying a logarithm to the final output to increase contrast in the spectrogram plot. All code is available in Appendix D.

First, the sound data needs to be loaded in. The sound data is given as a **.m4a** file. In order to access the raw sound data, the function `audioread` is used. This takes in the **.m4a** file and provides the signal values and the sampling rate, which uniquely characterize the signal. Dividing the number of data points in the signal by the sampling rate gives the length of the signal in seconds. A one-dimensional grid of points is then created that divides the total time of the signal by the number of data points. The distance between points is then equal to the sampling rate. The last time point is ignored since the signal is considered as periodic. This gives the time domain for the problem. The frequency domain is constructed by taking the number of data points minus one, scaling by $\frac{2\pi}{L}$ since the Fast Fourier Transform (FFT) algorithm expects a 2π periodic domain, and multiplying by the grid $[0 : -\frac{n}{2} - 1, -\frac{n}{2} : -1]$. This accounts for the shift resultant from the FFT algorithm. A shifted version `ks` is then created by applying `fftshift`, removing the earlier shift effect.

The two results of interest are the spectrogram and the list of maximum frequencies for each time step. Both are initialized as empty arrays. For each problem, there is a plausible range of frequencies. These frequencies must be defined to create corresponding high or low pass filters. Since the frequency domain is symmetric about zero, the negative frequencies are discarded for every problem. This defines a new, non-negative grid. This grid is then sliced to select only the plausible frequencies, acting as a filter. However, the plausible frequencies must be multiplied by $\frac{L}{2\pi}$ to convert them to the units used by `ks` indexing. This sliced grid should then have all of its values, not indices, divided by 2π, to convert from angular frequency to frequency.

3.1.2 Creating the Spectrogram

Unlike the FFT, the Gabor transform is applied individually to different intervals of the time domain. The Gabor transform itself acts by multiplying the signal by a Gaussian centered at a certain time, then taking the FFT of this filtered signal. The size of these intervals plays an important role in the transform. Since Gaussian's have long, diminishing tails, there will be overlap of the filter between each interval. The translation parameter `b` modulate the amount of overlap. More overlap comes at computational expense. Higher The width of the Gaussian, given by a parameter `a` will modulate the resolution of the frequency domain and time domain. A smaller `a` results in a wider Gaussian and better frequency resolution, and a larger `a` results in a thinner Gaussian and a better time resolution. Both of these parameters are critical to the results of the spectrogram. After experimentation, the translational parameter that was determined to provide the

best trade-off between increased resolution and manageable computational complexity was 0.05 in the first two problems and 0.01 in the third problem, the guitar score in *Comfortably Numb*. Similarly, the Gaussian width with the optimal width for temporal resolution and frequency resolution was found to be 250 in the first two problems and 2000 in the third problem.

After the filter is applied, the FFT is taken. To post-process this transform, the frequency-domain signal is shifted, the negative frequencies are removed, and only the plausible frequencies are filtered. The filtering is performed by slicing the frequency domain signal such that only plausible frequencies remain. The plausible frequencies should be multiplied by `L` first, to be in the proper units of ks. This signal is then added to the spectrogram. After all intervals have been transformed and added to the spectrogram, it is complete. There will be `L/b` points in the time dimension, rounded up to the nearest integer. For the case of the guitar solo in *Comfortably Numb*, the resultant spectrogram is amplified by taking `log(|Signal|+1)`, since the signal is otherwise relatively weaker than its background. The maximum frequency at each interval point is also recorded. This will give a heuristic for what note is being played, though human interpretation is the most important factor in determining the score.

3.1.3 Presenting the Results

The spectrogram is then plotted in a two-dimensional pseudocolor graph, with the amplitude represented as a color. The color scheme is designed to resemble heat, to be more humanly readable. The x-axis is time and the y-axis is frequency. The maximum frequencies of the signal are also visualized below as a line-plot connecting the highest frequencies at each interval. The frequencies of the twelve notes are added to the graph as horizontal lines, to allow for easy reference of which note is represented by the frequencies on each graph.

The function `note_map`, shown in Algorithm 2 maps each frequency in the `max_freq` list to its corresponding note. Repeated maximum frequencies are not included, since they likely correspond to the same note being played. The function includes a `proximity_threshold` parameters that determines how close a frequency must be to a note's frequency to determine that the frequency corresponds to that note. It was set to 2 Hz. This is also only meant as an additional heuristic for human interpretation.

4 Computational Results

All figures are given in Appendix A. Figure 1 shows the results for the *Sweet Child 'O Mine* intro. Figure 2 shows the results for the bass of the *Comfortably Numb* solo. Figure 3 shows the results for the guitar of the *Comfortably Numb* solo. All three of these show the spectrograms of the songs, with Figure 2 showing a low-pass filter for the plausible bass notes and Figure 3 showing a high-pass filter for the plausible guitar notes. Figure 4 shows a zoomed in version of Figure 1 to show how the horizontal note lines allow for the identification of the note being played, and how the maximum frequency list picks this up. Figure 5 shows a zoomed in version of Figure 2, showing how it becomes humanly readable with the note-frequency horizontal lines. Figure 6 shows a zoomed in version of the *Comfortably Numb* guitar spectrogram, showing how individual notes can be identified, though with considerably more difficulty.

The `note_map` function provides a list of the notes in each song. These can be found in Appendix C. While these represent a part of the entire picture, it is important to note that the maximum frequency graph and `note_map` output are only heuristics. For example, it is significantly easier to read off the actual score of the bass in *Comfortably Numb*: B2 A2 G2 F# E2. A zoom-in of that was used for this determination can be seen in Figure 5. While these values can be seen in the notes mapping, this is more for verification of the human interpretation of the plot than the definitive answer. In fact, it is immediately clear that the `note_map` procedure is not reliable. Selecting for maximum frequency still introduces a significant amount of noise. As such, while interesting potential tools for analyzing the musical score, they should not be used. However, in trying to use them some of the difficulties of signal analysis have been highlighted.

The Pink Floyd guitar solo represents another challenge. Whereas the *Sweet Child O' Mine* guitar and

Comfortably Numb bass parts are relatively clear, the guitar solo for *Comfortably Numb* is not. Upon determining that the maximum frequency outputs and notes mapping were not reliable for even the easier tasks, they will not be used here. Human interpretation will be relied upon. The requiring of human interpretation does not diminish the importance of the spectrogram, as it still facilitates intuitive read-offs of the frequencies from the graph. An example of a close-up of this spectrogram is shown in Figure 6, whose first few notes are F#4 F#4 D4 D5 A4 E5 E4 A4 D5 E4 A4 D5 E4. This represents a best guess, and it is noted there are likely errors involved in this process.

The actual scores, using heuristics and human interpretation, are:

Sweet Child of Mine

C#3 C#4 G#4 F#4 F#5 G#4 F5 G#4 C#3 C#4 G#4 F#4 F#5 G#4 F5 G#4 D#3 C#4 G#4 F#4 G#4 F5 G#4 D#3 C#4 G#4 F#4 F#5 G#4 F5 G#4 F#4 C#4 G#4 F#4 G#4 F5 G#4 F#4 C#4 G#4 F#4 F#5 G#4 F5 G#4 C#3 C#4 G#4 F#4 F#5 G#4 F5 G#4 C#3

Comfortably Numb (Bass)

(B2 A2 G2 F# E2) (Repeat 3x) B2

There are also potential repeats among the first few notes. For instance, it appears A2 is played at least twice, even though it is the same note.

Comfortably Numb (Guitar)

This was only analyzed for one segment, comprising 11.886 seconds, or approximately one-fifth of the solo. It is considered that this gives the flavor of the analysis, while providing a smaller test case with which it could be scrutinized in detail. Additionally, it is expected that the transcription is of relatively low fidelity, a result not expected to improve if more of the solo was analyzed.

F#4 F#4 D4 D5 A4 E5 E4 A4 D5 E4 A4 D5 E4 (Pause) (E3 bend to F#4) D5 F#3 D5 D5 D5 F5 F#5 B4 F#4 B3

5 Summary and Conclusions

The Gabor transform was used on two segments of music, facilitating a computational approach to the process the human nervous system undertakes naturally. This allowed for a sound signal in the time domain to be taken in, and with minor human interpretation, the musical score to be determined. However, there were limitations to this procedure. Minimal human intervention was still needed, with two approaches toward automation failing. Additionally, even human intervention was not able to transcribe the notes of a more complex melody with high fidelity. There were still many upsides to the technique. Filtering was easy to implement, allowing the separation of musical instruments by their frequency domains. After setup, there were only two parameters to set, the Gaussian width and the translational parameter, not including the high and low values of any filter. The greatest benefit of the Gabor transform in this setting was in transforming a completely uninterpretable signal into a new data structure that had clear interpretation. This also has strong potential as a preprocessing step for data with latent frequency domain information.

References

[1] *Gabor Limit*. URL: `https://en.wikipedia.org/wiki/Gabor_limit`.

[2] *Guitar*. URL: `https://en.wikipedia.org/wiki/Guitar`.

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

Appendix A Figures

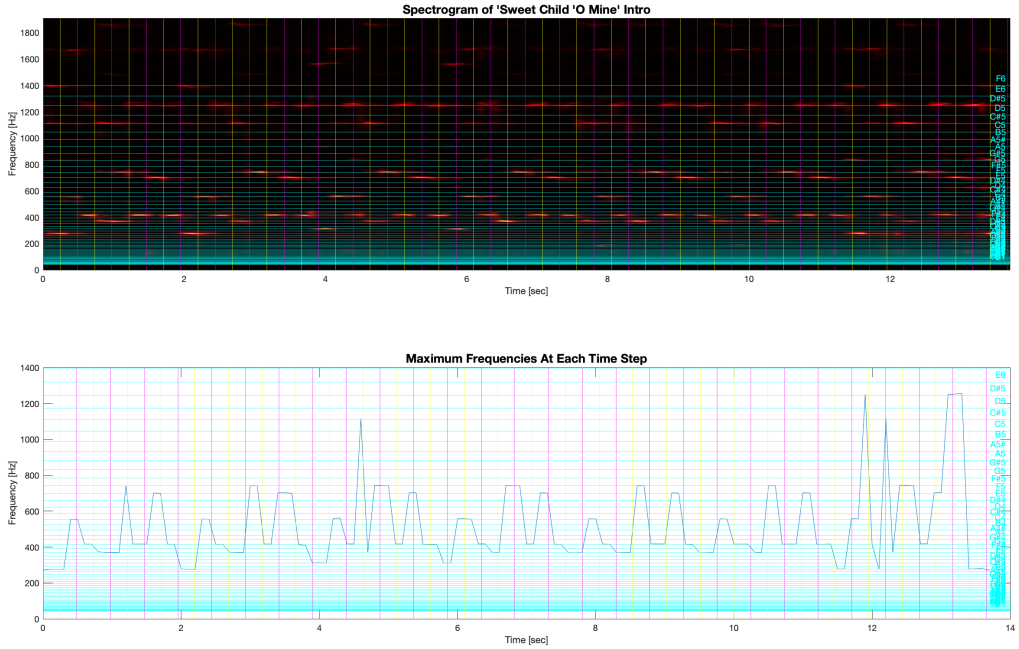

Figure 1: (Top) The spectrogram from the Sweet Child O' Mine Intro with parameters `a = 250` and `b = 0.05`. Filtered for positive frequencies. (Bottom) The maximum frequencies from the spectrogram above. Similar to a 'connect the dots' of the brightest points on the spectrogram at each time slice.

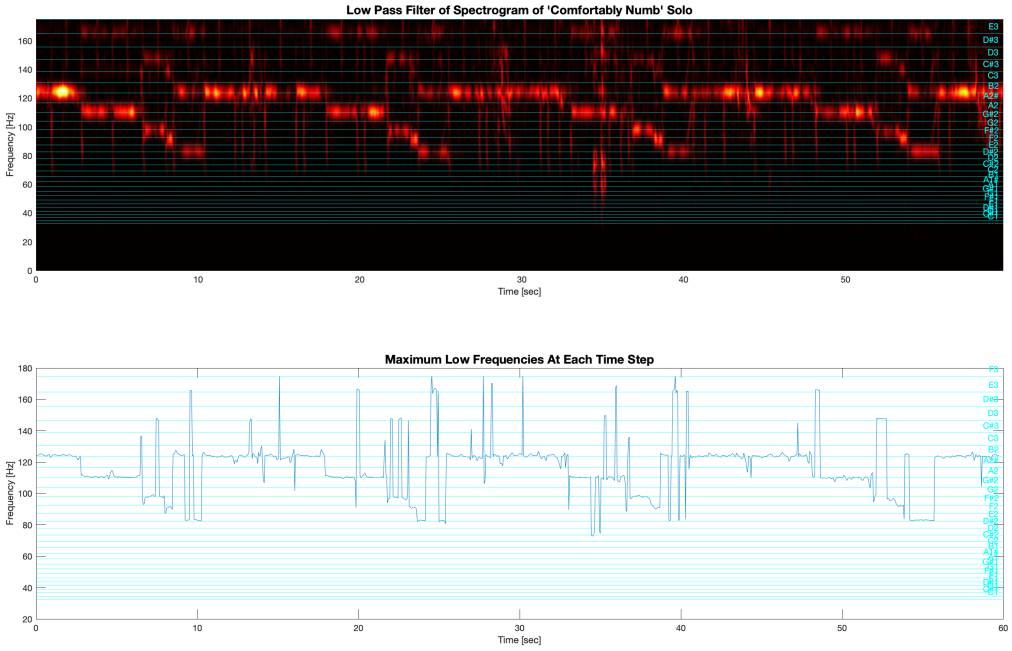

Figure 2: (Top) The spectrogram from the Comfortably Numb solo with parameters `a = 250` and `b = 0.05`. Filtered for positive frequencies and plausible bass frequencies. (Bottom) The maximum frequencies from the spectrogram above.

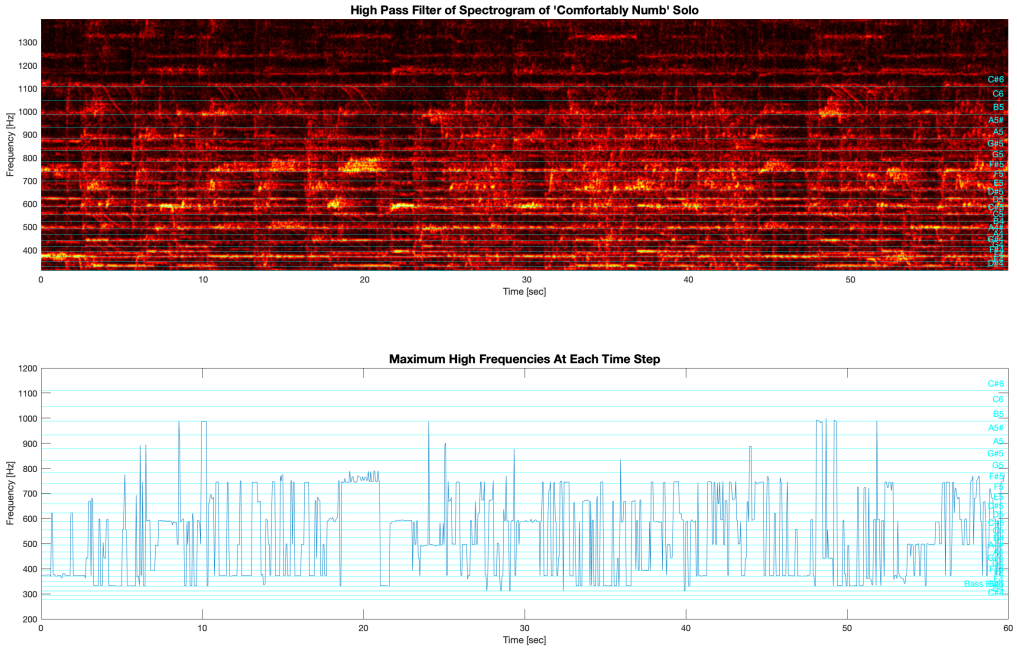

Figure 3: (Top) The spectrogram from the Comfortably Numb solo with parameters `a = 2000` and `b = 0.01`. Filtered for positive frequencies and plausible guitar frequencies. (Bottom) The maximum frequencies from the spectrogram above.

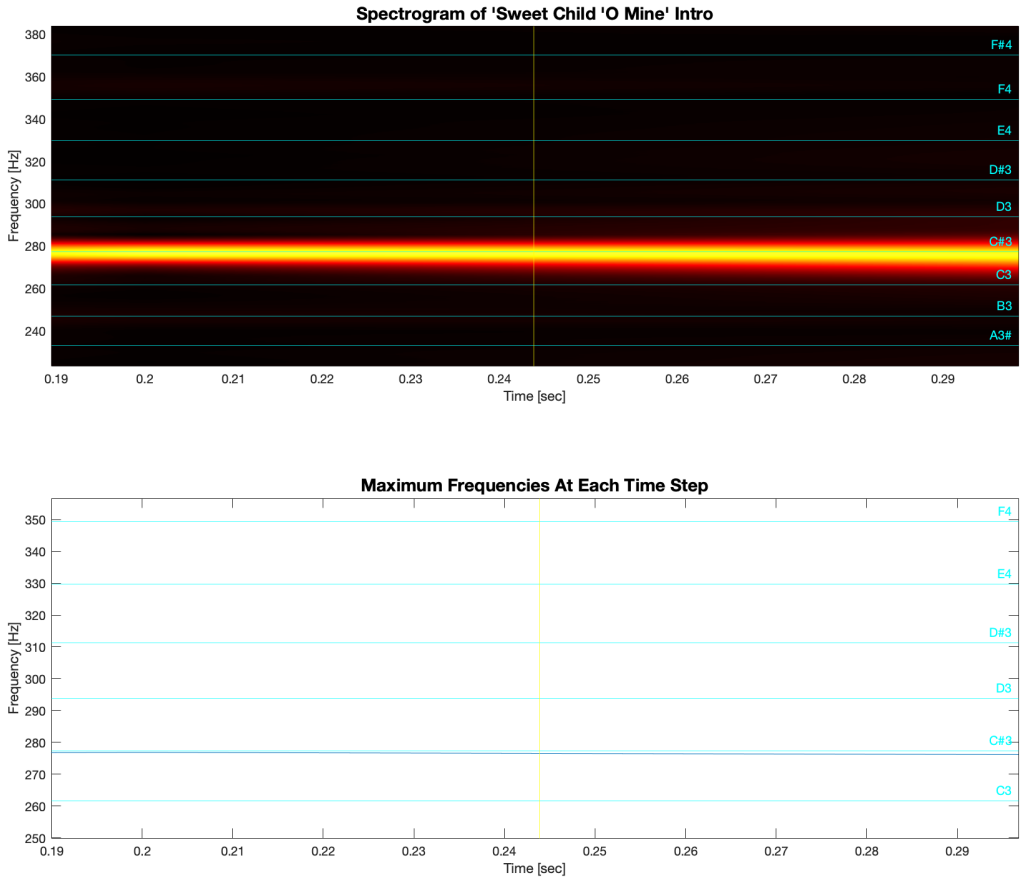

Figure 4: A zoomed in version of the Sweet Child O' Mine spectrogram to see how the horizontal overlay of the note-frequencies allows for easy visualization of the note being played at each time point.

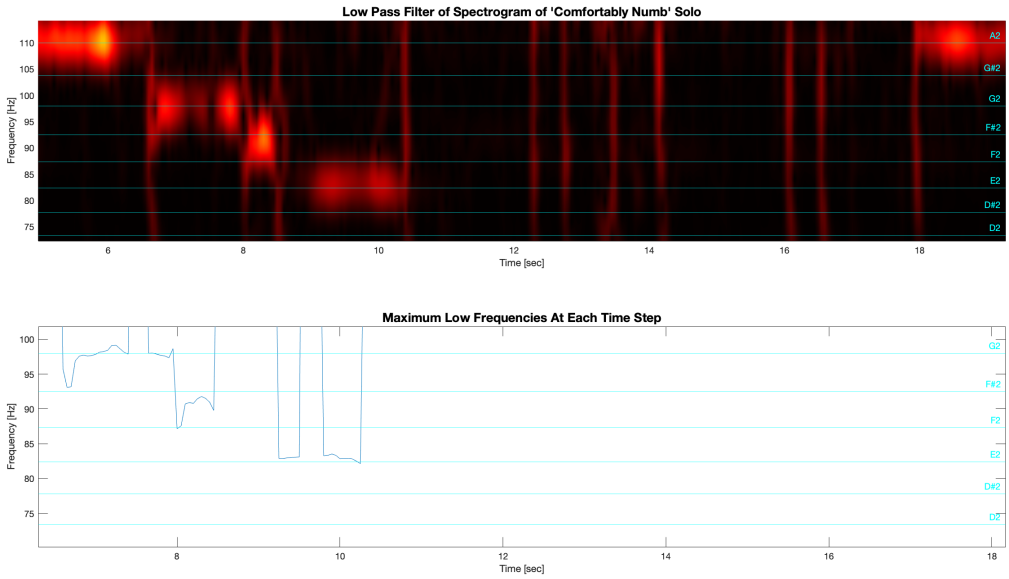

Figure 5: A zoomed in version of the Comfortably Numb bass filter spectrogram to see how the notes can be determined.

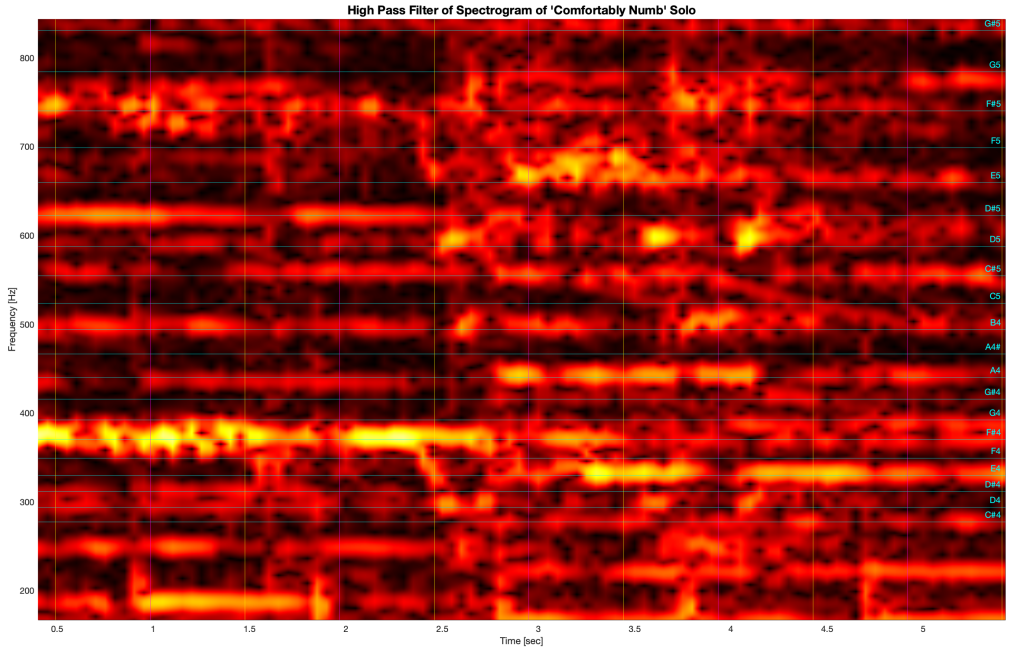

Figure 6: A zoomed in version of the Comfortably Numb guitar filter spectrogram to see how the notes can be somewhat determined.

Appendix B MATLAB Functions

- `[y,Fs] = audioread(filename)` reads data from an audiofile named `filename`, 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`.
- `Y = fft(X)` is the the Fast Fourier Transform. `X` is data in the spatial domain and `Y` is data in the frequency domain. There is a shift and sign-flip associated with the FFT algorithm in each dimension.
- `Y = ifft(X)`is the inveres Fourier Transform. It takes in a shifted and sign-flipped Fourier transformed signal.
- `Y = fftshift(X)`undoes the shifting associated with the FFT algorithm.
- `M = max(A)` finds the maximum value of an array `A`. Importantly, if a second output is provided, in the form `[max, ind] = max(input)`, then the `ind` returns the index of the maximum value. However, only one value is returned for the index. If `input` has a dimension higher than one, the index is returned using the MATLAB convention, not as multidimensional subscripts. If only the index is needed, a `~` can be used as a placeholder for the `max`.
- `pcolor(C)` creates a psuedocolor plot based on a matrix `C`. Since the matrix is two-dimensional, the plot will be two-dimensional with a color given to each position based upon the value in the matrix. Use `shading interp;` afterward to smooth the colors.
- `yline(yvalue)` creates a horizontal line on a plot at `yvalue`.
- `xline(xvalue)` creates a vertical line on a plot at `xvalue`.

Appendix C Note Mappings

The `note_map` function provides a liist of the notes in each song.

For *Sweet Child O' Mine* the result was:
C#3 C#4 G#4 F#4 F#5 G#4 F5 G#4 C#3 C#4 G#4 F#4 F#5 G#4 F5 D#3 G#4 F#4 F#5 G#4 D#3 C#4 G#4 F#4 F#5 G#4 F#4 C#4 G#4 F#4 F#5 G#4 F#4 C#4 G#4 F#4 F#5 G#4 C#3 C#4 G#4 C#3 F#4 C#3

For *Comfortably Numb* the bass result was:
B2 A2 A#2 A2 C#2 F#2 G2 D2 G2 F2 F#2 B2 E2 E3 B2 E2 B2 D2 B2 C2 A2 B2 F3 B2 G#2 B2 A2 F#2 E3 A2 B2 A2 F#2 G2 D2 G2 D2 G2 D2 F#2 E2 B2 E3 F3 E3 B2 E3 E2 B2 F3 B2 A#2 B2 F3 B2 A#2 B2 A#2 B2 A2 D1 D#1 A2 D#1 D1 A2 D2 A2 F2 G2 F#2 B2 E2 B2 E3 F3 E3 E2 B2 E2 E3 A#2 B2 A2 B2 D2 B2 A2 E3 A2 G#2 A2 F#2 G2 D2 G2 F#2 E2 B2 E2 B2 G#2 B2

As discussed in the text, these do not end up being particularly useful. As such, the notes for the *Comfortably Numb* guitar solo, which would also be significantly longer, is omitted.

Appendix D Appendix C: MATLAB Code

```
num_freqs = [0:61];
freq_list = 41.21 .* (2^(1/12)).^num_freqs;
name_list = ["E1","F1","F#1","G1","G#1","A1","A1#","B1","C1","C#1","D1","D#1",...
             "E2","F2","F#2","G2","G#2","A2","A2#","B2","C2","C#2","D2","D#2",...
             "E3","F3","F#3","G3","G#3","A3","A3#","B3","C3","C#3","D3","D#3",...
             "E4","F4","F#4","G4","G#4","A4","A4#","B4","C4","C#4","D4","D#4",...
             "E5","F5","F#5","G5","G#5","A5","A5#","B5","C5","C#5","D5","D#5",...
             "E6","F6"];
length(freq_list)
length(name_list)

function f = note_map(max_freq)
    f = [];
    proximity_threshold = 2;
    for i=1:length(max_freq)
        for j=1:length(freq_list)
            if abs(max_freq(i) - freq_list(j)) < proximity_threshold
                if length(f) == 0
                    f = [f; name_list(j)];
                elseif (f(length(f)) ~= name_list(j))
                    f = [f; name_list(j)];
                end
            end
        end
    end
end
```

Listing 1: MATLAB code used to convert frequencies to notes, available at https://github.com/benfrancis314/AMATH582

```
clear all; close all; clc;
[y, Fs] = audioread('GNR.m4a');
%% Setup
S = y'; n = length(S); % Number of data points in sample chunk
L = n/Fs; % record time in seconds. This is time of *chunk*, not entire recording
t2 = linspace(0,L,n+1); t = t2(1:n);
k = (2*pi/L)*[0:n/2-1, -n/2:-1]; ks = fftshift(k);
Sgt_spec = []; max_freq = [];% Setup emptry spectrogram and maximum frequency list
zero_freq_ind = length(ks)/2+1; % Start at lowest freq possible
ks_cut = ks(zero_freq_ind:length(ks));
plausible_freq_max = 12000;
ks_plausible = ks_cut(1:floor(plausible_freq_max*L/(2*pi))) / (2*pi);
%% Perform Gabor Transform
filter_width = 250; translation_jump = 0.1;
tslide = 0:translation_jump:L;
for i=1:length(tslide)
    g=exp(-filter_width*(t-tslide(i)).^2); % Define Gabor filter
    Sg = g.*S; % Apply filter to signal
    Sgt = fft(Sg); % FFT the filtered signal
    Sgt_abs = abs(fftshift(Sgt)); Sgt_cut = Sgt_abs(zero_freq_ind:length(ks));
    Sgt_plausible = Sgt_cut(1:floor(plausible_freq_max*L/(2*pi)));
    Sgt_spec = [Sgt_spec; Sgt_plausible]; % Add to spectrogram
    % Determine max frequency
    [~, max_freq_sample] = max(Sgt_plausible);
    max_freq = [max_freq; ks_plausible(max_freq_sample)];
end
%% Presentation
num_freqs = 0:61; freq_list = 41.21 .* (2^(1/12)).^num_freqs;
name_list = ["E1","F1","F#1","G1","G#1","A1","A1#","B1","C1","C#1","D1","D#1",...
             "E2","F2","F#2","G2","G#2","A2","A2#","B2","C2","C#2","D2","D#2",...
             "E3","F3","F#3","G3","G#3","A3","A3#","B3","C3","C#3","D3","D#3",...
             "E4","F4","F#4","G4","G#4","A4","A4#","B4","C4","C#4","D4","D#4",...
             "E5","F5","F#5","G5","G#5","A5","A5#","B5","C5","C#5","D5","D#5",...
             "E6","F6"];
bpm = 123; bps = 60/bpm;
figure(1); subplot(2,1,1), pcolor(tslide,ks_plausible,Sgt_spec.'), shading interp; hold on;
set(gca,'Ylim', [0 max(ks_plausible)]);
colormap(hot), title("Spectrogram of 'Sweet Child 'O Mine' Intro", 'FontSize',14), xlabel('Time [sec]')
for i=1:length(name_list) % Add horizontal lines for each note
    yline(freq_list(i),'c',name_list(i));
end
for b=0:floor(L/bps) % Add vertical lines for each beat
    xline(bps*b,'m'); xline(bps*b+(bps/2),'y');
end
subplot(2,1,2), plot(tslide,max_freq);  % Plot maximum frequencies
title("Maximum Frequencies At Each Time Step", 'FontSize',14), xlabel('Time [sec]'), ylabel('Frequency
for i=1:length(name_list) % Add horizontal lines for each note
    yline(freq_list(i),'c',name_list(i));
end
for b=0:floor(L/bps) % Add vertical lines for each beat
    xline(bps*b,'m'); xline(bps*b+(bps/2),'y')
end
note_map(max_freq)
```

Listing 2: MATLAB code used to create spectrogram for Sweet Child O' Mine, available at https://github.com/benfrancis314/AMATH582

```
clear all; close all; clc;
[y, Fs] = audioread('Floyd.m4a');
%% Setup
S = y'; n = length(S); % Number of data points in sample chunk
L = n/Fs; % record time in seconds
t2 = linspace(0,L,n+1); t = t2(1:n);
k = (2*pi/L)*[0:n/2-1, -n/2:-1]; ks = fftshift(k);
St = fft(S); % FFT of signal
Sgt_spec_bass = []; max_bass = []; % Setup emptry spectrogram and max frequency list for bass
zero_freq_ind = length(ks)/2+1; ks_cut = ks(zero_freq_ind:length(ks));
plausible_bass_high = 300; plausible_freq_min = 1/L; plausible_freq_max = 1400;
ks_bass = ks_cut(floor(plausible_freq_min*L):floor(plausible_bass_high*L))/(2*pi);
%% Perform Gabor Transform
filter_width = 250; translation_jump = 0.01; tslide = 0:translation_jump:L;
for i=1:length(tslide)
    g=exp(-filter_width*(t-tslide(i)).^2); % Define Gabor filter
    Sg = g.*S; % Apply filter to signal
    Sgt = fft(Sg); % FFT the filtered signal
    Sgt_abs = abs(fftshift(Sgt)); Sgt_cut = Sgt_abs(zero_freq_ind:length(ks));
    Sgt_bass_plausible = Sgt_cut(floor(plausible_freq_min*L):floor(plausible_bass_high*L));
    Sgt_spec_bass = [Sgt_spec_bass; Sgt_bass_plausible]; % Add to spectrogram
    [~, max_bass_sample] = max(Sgt_bass_plausible);
    max_bass = [max_bass; ks_bass(max_bass_sample)];
end
%% PRESENTATION
num_freqs = 0:61;freq_list = 32.7 .* (2^(1/12)).^num_freqs; % Music Notes Setup
name_list = ["C1","C#1","D1","D#1","E1","F1","F#1","G1","G#1","A1","A1#","B1",...
             "C2","C#2","D2","D#2","E2","F2","F#2","G2","G#2","A2","A2#","B2",...
             "C3","C#3","D3","D#3","E3","F3","F#3","G3","G#3","A3","A3#","B3",...
             "C4","C#4","D4","D#4","E4","F4","F#4","G4","G#4","A4","A4#","B4",...
             "C5","C#5","D5","D#5","E5","F5","F#5","G5","G#5","A5","A5#","B5",...
             "C6","C#6"];
bpm = 61; bps = 60/bpm;
figure(1); subplot(2,1,1), pcolor(tslide,ks_bass,Sgt_spec_bass.'), shading interp;
set(gca,'Ylim', [0 max(ks_bass)]);
colormap(hot), title("Low Pass Filter of Spectrogram of 'Comfortably Numb' Solo", ...
    'FontSize',14), xlabel('Time [sec]'), ylabel('Frequency [Hz]');
yline(800,'c','Bass High');
for n=1:30
    yline(freq_list(n),'c',name_list(n));
end
for b=0:floor(L/bps) % Add vertical lines for each beat
    xline(bps*b,'m'); xline(bps*b+(bps/2),'y')
end
subplot(2,1,2), plot(tslide,max_bass);
title("Maximum Low Frequencies At Each Time Step", 'FontSize',14), xlabel('Time [sec]'), ylabel('Freque
for n=1:30
    yline(freq_list(n),'c',name_list(n));
end
for b=0:floor(L/bps) % Add vertical lines for each beat
    xline(bps*b,'m'); xline(bps*b+(bps/2),'y')
end
bass_notes = note_map(max_bass)
```

Listing 3: MATLAB code used to create spectrogram for Comfortably Numb with filtered bass, available at https://github.com/benfrancis314/AMATH582

```
clear all; close all; clc;
[y, Fs] = audioread('Floyd.m4a');
%% Setup
S = y'; n = length(S); % Number of data points
L = n/Fs; % record time in seconds
t2 = linspace(0,L,n+1); t = t2(1:n);
k = (2*pi/L)*[0:n/2-1, -n/2:-1]; ks = fftshift(k);
St = fft(S); % FFT of signal
Sgt_spec_guitar = []; % Setup emptry spectrogram for guitar
max_guitar = []; % maximum frequency for guitar
zero_freq_ind = length(ks)/2+1;
ks_cut = ks(zero_freq_ind:length(ks));
plausible_guitar_low = 164.81; plausible_freq_min = 65.41; plausible_freq_max = 1760;
ks_guitar = ks_cut(floor(plausible_guitar_low*L):floor(plausible_freq_max*L))/(2*pi);
%% Perform Gabor Transform
filter_width = 250; translation_jump = 0.05; tslide = 0:translation_jump:L;
for i=1:length(tslide)
    g=exp(-filter_width*(t-tslide(i)).^2);
    Sg = g.*S; Sgt = fft(Sg);
    Sgt_abs = abs(fftshift(Sgt)); Sgt_cut = Sgt_abs(zero_freq_ind:length(ks));
    Sgt_guitar_plausible = Sgt_cut(floor(plausible_guitar_low*L):floor(plausible_freq_max*L));
    Sgt_spec_guitar = [Sgt_spec_guitar; Sgt_guitar_plausible]; % Add to spectrogram
    [~, max_guitar_sample] = max(Sgt_guitar_plausible);
    max_guitar = [max_guitar; ks_guitar(max_guitar_sample)];
end
%% Presentation
num_freqs = 0:49; freq_list = 65.41 .* (2^(1/12)).^num_freqs; % Music Notes Setup
name_list = ["C2","C#2","D2","D#2","E2","F2","F#2","G2","G#2","A2","A#2","B2",...
             "C3","C#3","D3","D#3","E3","F3","F#3","G3","G#3","A3","A#3","B3",...
             "C4","C#4","D4","D#4","E4","F4","F#4","G4","G#4","A4","A#4","B4",...
             "C5","C#5","D5","D#5","E5","F5","F#5","G5","G#5","A5","A#5","B5",...
             "C6","C#6"];
bpm = 61; bps = 60/bpm;
figure(1); subplot(2,1,1),pcolor(tslide,ks_guitar,log(abs(Sgt_spec_guitar.')+1)), shading interp;
set(gca,'Ylim', [min(ks_guitar) max(ks_guitar)]);
colormap(hot), title("High Pass Filter of Spectrogram of 'Comfortably Numb' Solo", 'FontSize',14), ...
    xlabel('Time [sec]'), ylabel('Frequency [Hz]');
for n=1:length(name_list)
    yline(freq_list(n),'c',name_list(n));
end
for b=0:floor(L/bps)
    xline(bps*b,'m'); xline(bps*b+(bps/2),'y')
end
subplot(2,1,2), plot(tslide,max_guitar);
title("Maximum High Frequencies At Each Time Step",'FontSize',14), xlabel('Time [sec]'), ylabel('Freque
yline(311.13,'c','Bass High');
for n=26:length(name_list)
    yline(freq_list(n),'c',name_list(n));
end
for b=0:floor(L/bps)
    xline(bps*b,'m'); xline(bps*b+(bps/2),'y')
end
guitar_notes = note_map(max_guitar)
```

Listing 4: MATLAB code used to create spectrogram for Comfortably Numb with filtered guitar, available at https://github.com/benfrancis314/AMATH582