Utilizing the Gabor Transform for Music Composition

Carlos Sama 10, Feb. 2021

Abstract

This report provides an application of the Gabor Transform to perform frequency analysis on popular rock songs. The analysis, performed in MATLAB, includes the creation of a spectrograms for Guns n' Roses, "Sweet Child O' Mine," and for Pink Floyd's "Comfortably Numb." Additionally, instrumental tracks for each of the songs are recreated by quantizing measured frequencies from the spectrograms.

Introduction

With regards to the processing a signal, the Fourier Transform is unable to provide any time/location-based knowledge. Instead, it encodes total information of a signal's frequency components and their significance. But in some applications of signal processing, for example, analyzing a piece of music, it is necessary to know when certain frequencies occur—because that's how music works. The Gabor Transform can be used to uncover some information regarding when a frequency occurs, and some information about what that frequency specifically is. Yet performing this analysis on a piece of music will reveal highly complex frequency-time information as the layering of musical instruments give many overlapping frequencies. Furthermore, every musical instrument has overtones, which are several other small amplitude frequencies on top of the frequency of the note itself. For example, a guitar playing an A 440 may have various, higher, random frequencies detectable but small in magnitude, thus giving the guitar its unique sound. This phenomenon is depicted in Figure 1. [1]

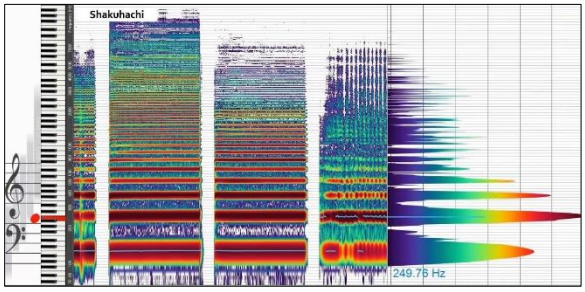

Figure 1 Overtone frequencies of various instruments when playing the same 'C' musical note [1]

Theoretical Background

The Gabor Transform works by applying the Fourier Transform to a window that traverses the entire signal. The window used in this application is a gaussian distribution function. This function gives the ability to widen or shrink the size of the window being traversed. A wider signal will provide for better frequency information as more of the original signal is included but have less precision regarding the time of discovered frequencies as its wider base captures more time. The equation for a continuous representation of the Gabor Transform is shown by equation 1, where g is the window function and f is the function being transposed.

(1)

$$F(m,n) = \int_{-\infty}^{\infty} f(t)\, g(t - nt_0)\, e^{2\pi i m \omega t} dt$$

For this application, a discretized version of the Gabor Transform is implemented. In the discrete case, the rate at which this window traverses the function is defined with great importance. A step length must be chosen that is smaller than the window itself to ensure overlapping of each window occurs.

Lastly, the information from the Gabor Transform can be used to recreate specific parts of a song. For example, by identifying the most significant low frequency components at each step in the song, a musical score can be arranged for the part.

Algorithm Implementation

The defined analysis method was implemented in MATLAB. Short sections of “Sweet Child O’ Mine,” and “Comfortably Numb” were loaded in as sound amplitude vectors with a sample frequency of 48000 Hz and 44100 Hz, respectively. The Discrete Gabor Transform was applied to the sound vectors using a gaussian distribution window applied every 0.1 seconds. Frequency components from each window was saved in a matrix to be plotted as a spectrogram.

From the spectrograms, frequency ranges for the guitar and bass parts of each song were determined by inspection. Generally, the guitar range was found to span from 1000-3000, and the base from 300-1000. These ranges were used to define filters which isolated the instrument’s part in the frequency domain. The isolation filters are indicated in equation 2.

(2)

$$f = \begin{Bmatrix} x < \omega_L, & 0 \\ \omega_L < x < \omega_H, & 1 \\ \omega_H < x, & 0 \end{Bmatrix}$$

From the isolated signal, the largest frequency component at each time step was found, and that frequency value was scaled to proper time units and quantized to the nearest corresponding note on the A440 musical scale [2]. The repeated frequencies were grouped to create a musical score of each note played—omitting time signatures and specifications on how long to hold each note.

To verify the transcribed score, the identified notes were transformed into a sound vector which could be played. A sine wave of the identified musical frequency was created and extended for the duration that note was held. The sine waves for each note were concatenated to form the overall vector and then replayed using the initial sample frequency.

Results

Applying the Gabor Transform to the provided sound samples yielded a matrix which could be plotted as a spectrogram. The spectrograms from “Comfortably Numb,” and “Sweet Child O’ Mine” are shown in Figure 2.

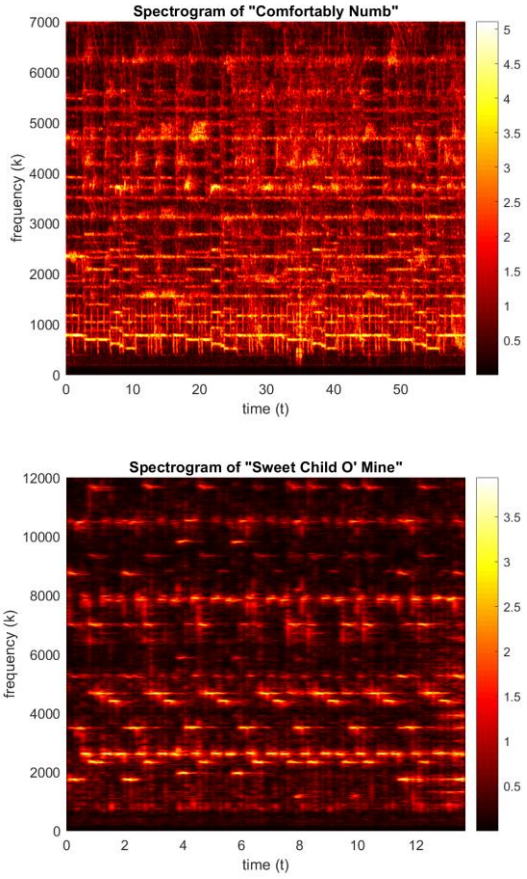

Figure 2 (Top) Spectrogram of "Comfortably Numb" (Bottom) Spectrogram of "Sweet Child O' Mine"

The spectrograms above depict all the discovered frequency components, giving a very messy image, and some of the components are even outside the A440 range. Many of these frequencies that appear are due to the

overtones produced by an instruments sound. For example, the notes for the guitar solo in “Sweet Child O’ Mine” are contained in the lower band from around 1500-5000; however, many more distinct components can be seen above that range. This is because the distortion of the electric guitar layers many high frequency perturbations on top of the original note. Additionally, the spectrogram is displaying frequencies from every instrument at that window in time. For example, in the “Comfortably Numb” chart, the baseline can be seen between 500-1000, and the guitar/layered instruments can be seen from 1000-2500. To verify this, a filter can be applied to the overall Fourier Transform of the sound signal with a filter around the desired sound range. In this case, a gaussian filter centered at 750 was applied to achieve just the bass noise in comfortably numb. The resulting spectrogram and sound vector is plotted in Figure 3.

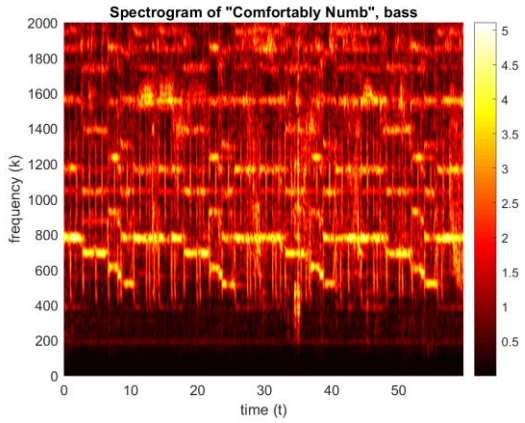

Figure 3 "Comfortably Numb" Spectrogram of bass

The isolated guitar frequency from “Sweet Child O’ Mine,” and bass frequency from “Comfortably Numb,” once quantized to A 440, produce the melodies in detailed in Tables 1 and 2 respectively in Appendix A. Lastly, the guitar solo from comfortably numb was targeted from the Gabor Transform; however, the overtones and harmonics made parsing out specific notes very challenging. The resulting guitar solo is detailed in Table 3 of Appendix A.

Conclusion

By applying the Gabor Transform to sound vectors of popular rock songs, individual instrument melodies can be extracted and recreated. The Gabor Transform enables this functionality by visually representing the frequency space over time while maintaining enough information needed to filter out harmonics and overtones.

Appendix A

Utilized MATLAB Functions:

`fft:` This function performs the Discrete Fast Fourier Transform on a provided vector and returns the calculated frequency components in the same dimensioned vector.

`ifft:` This function performs the Inverse Discrete Fast Fourier Transform on a provided vector and returns the calculated signal in the same dimensioned vector.

`fftshift:` This function takes a provided vector of frequency data and shifts the negative frequency components to the left of the zero so that the components are arranged in ascending order.

`max:` This function returns the maximum value and its index of a provided dataset.

`audioplayer:` This is a class which contains a sound file consisting of a sound vector and the sample frequency to play the sound at.

`playblocking:` This function uses the computer’s speakers to play the provided audioplayer object.

pcolor: This function produces a “checkerboard” color plot of the provided array data tabulated by provided x and y axis breakpoints. Various color patterns can be used to delineate intensity. For this case, the “hot” color pattern was used.

Sweet Child O' Mine Transcribed Guitar Solo

Table 1. "Sweet Child O' Mine" notes as listed on the A440 scale in [2]

D5 | D6 | A5 | G5 | A5 | F♯6/G♭6 | A5 | D5 | D6 | A5 | G5 | G6 | A5 | F♯6/G♭6 | A5 | E5 | D6 | A5 | G5 | G6 | A5 | F♯6/G♭6 | A5 | E5 | D6 | A5 | G5 | G6 | A5 | F♯6/G♭6 | A5 | G5 | D6 | A5 | G5 | G6 | A5 | F♯6/G♭6 | A5 | G5 | D6 | A5 | G5 | G6 | A5 | F♯6/G♭6 | A5 | D5 | D6 | A5 | G5 | G6 | A5 | F♯6/G♭6 | A5 | D5

Comfortably Numb Transcribed Bass Line

Table 2. "Comfortably Numb" notes as listed on the A440 scale in [2]

| B2 | A2 | G2 | F♯2/G♭2 | E2 | B2 | A2 | G2 | F♯2/G♭2 | E2 | B2 | A2 | G2 | F♯2/G♭2 | E2 | B2 | A2 | G2 | F♯2/G♭2 | E2 | B2 |

Comfortably Numb Transcribed Guitar Solo

| F♯4/G♭4 | F♯3/G♭3 | F♯4/G♭4 | E4 | E4 | E3 | E4 | G4 | D5 | D5 | E3 | B3 | F♯4/G♭4 | B3 | B3 | E5 | B3 | F♯4/G♭4 | B3 | F♯3/G♭3 | D5 | E3 | F♯5/G♭5 | F♯5/G♭5 | E3 | E4 | D5 | B4 | E3 | F♯3/G♭3 | F♯3/G♭3 | F♯3/G♭3 | D5 | F♯3/G♭3 | D5 | D4 | F♯3/G♭3 | B3 | F♯5/G♭5 | E4 | E4 | E4 | E4 | G4 | G3 | D5 | B4 | E3 | D5 | F♯3/G♭3 | B3 | F♯3/G♭3 | B3 | F♯5/G♭5 | B3 | F♯3/G♭3 | B3 | E3 | A4 A440 | E4 | E3 | B3 | A3 | D5 | F♯5/G♭5 | G3 | B4 | E3 | B4 | B3 | F♯3/G♭3 | B3 |

Appendix B

MATLAB CODE

```
clear all; close all; clc
%% Playing the song

% figure(1)
%[y, Fs] = audioread('GNR.m4a');
[y, Fs] = audioread('Floyd.m4a');
tr_gnr = length(y)/Fs; % record time in seconds
% plot((1:length(y))/Fs,y);
% xlabel('Time [sec]'); ylabel('Amplitude');
% title("Sweet Child O' Mine");
%p8 = audioplayer(y,Fs); playblocking(p8);

%% Taking the Gabor Transform

% Parameters
t = (1:length(y))/Fs;
L = max(t);
n = length(t);
k = (2*pi/L)*[0:n/2-1 -n/2:-1];
tau = 0:0.1:L;
alpha = 100;
ks = fftshift(k);

% Making sure my computer doesn't explode
easeUp = 10;
takeItEasy = zeros(length(y),1);
takeItEasy(1:easeUp:length(takeItEasy)) = 1;
Sgt_spec = zeros(length(find(takeItEasy)),length(tau));

% Performing the Gabor Transform
for j = 1:length(tau)
    Sgtf = zeros(length(y),1);
    g = exp(-alpha*(t - tau(j)).^2);
    Sg = g.*y';
    Sgt = fft(Sg);
    Sgts = fftshift(abs(Sgt))';
    Sgts = Sgts(logical(takeItEasy));
    Sgt_spec(:,j) = Sgts;

end

%% Plotting the Spectogram

figure
pcolor(tau,ks(logical(takeItEasy)),log(Sgt_spec+1))
shading interp
colormap(hot)
ylim([0, 1.2e4])
colorbar
xlabel('time (t)'), ylabel('frequency (k)')

%% Reproducing the Song

t = (1:length(y))/Fs;
L = max(t);
n = length(t);
k = (2*pi/L)*[0:n/2-1 -n/2:-1];
tau = 0:0.1:L;
alpha = 100000;
trbBand = [200, 1000];
song = zeros(0);
freqs = zeros(length(tau),1);
kvals = zeros(length(tau),1);

for i = 1:length(tau)
    g = exp(-alpha*(t-tau(i)).^10);
    yf = g'.*y;
    yt = fft(yf);
    yt(k<trbBand(1)) = 0;
    yt(k>trbBand(2)) = 0;
    ind = ind2sub(n, find(yt == max(yt)));
    kc = k(ind)/3;
    kvals(i) = kc;
    [val, idx] = min(abs(kc - piano_freq));
    freqs(i) = piano_freq(idx);
    note = sin((2*pi)*freqs(i)*t);
    note = note(1:1:floor(length(y) / length(tau)));
    song = [song, note];
end
%% Generating the Score

i = 1;
songFreq = [];
notes = '';
while i <= length(freqs)
    freq_o = freqs(i);
    noteLen = 0;
    while (freqs(i) == freq_o) && (i <= length(freqs))
        noteLen = noteLen + 1;
        freq_o = freqs(i);
        i = i+1;
        if i > length(freqs)
            break  % sorry :(
        end
    end
    if noteLen >= 2

        [val, noteIdx] = min(abs(freq_o - piano_freq));
        if ~isempty(notes)
            if ~strcmp(notes{end}, piano_notes{noteIdx})
                notes = strrep(strcat(notes, "_|_",
piano_notes{noteIdx}),'_',' ');
            end
        else
             notes = strrep(strcat(notes, "_|_",
piano_notes{noteIdx}),'_',' ');
        end
    end
end



%% Playing the Song
```

```
p8 = audioplayer(song*5,Fs); playblocking(p8); %filtered
song


%% Isolating the bass


t = (1:length(y))/Fs;
L = max(t);
n = length(y);
k = (2*pi/L)*[0:n/2-1 -n/2:-1];
y = y(1:end-1);
yt = fft(y);

bassFilter = exp(-0.00002*((k - 650).^2))';
```

```
ytf = yt.*bassFilter;

yb = ifft(ytf);

%% Playing the Song

p8 = audioplayer(yb*10,Fs); playblocking(p8);
```

References

[1] "Timbre: why different instruments playing the same tone sound different" *YouTube*, uploaded by What Music Really is. 14, May 2014

[2] "Tuning." *Frequencies of Musical Notes, A4 = 440 Hz*, Michigan Tech University, pages.mtu.edu/~suits/notefreqs.html.