AMATH 482 Hw2 - Music Notes

Ofir Kedar

Febuary 10, 2021

Abstract

In this project we create music scores for musical audio files and break the files into their frequency components to be manipulated. This is all tackled with the use of the Gabor Transform, which is a time inclusive use of the Fourier Transform which projects signals to frequency space. We find that splitting the files into discrete portions of time through filtering allows us to compute the Fourier transform in discrete portions and track where each frequency occurs in time. We can use this information to separate the frequencies as desired.

1 Introduction and Overview

Given two audio files, we are asked to produce the music score of the guitar in the first file (Sweet Child O Mine) and the bass in the second file (Comfortably Numb). We use the Gabor transform to show the frequency output at each time. Then we are asked to filter around the guitar and bass in their respective files, and put together the guitar solo in Comfortably Numb. For this portion, using information from the the previous part we can filter the entire signal around the general frequency of the instrument of choice in frequency space before converting it back to time space.

2 Theoretical Background

2.1 Gabor Transform

The **Gabor Transform** is a time (t) dependent expansion on the Fourier Transform. Using filter function $\boldsymbol{g(t)}$, chosen for this project to be the Gaussian Filter (See Hw1 - Submarine for more on Gaussian Filter), it filters function $\boldsymbol{f(t)}$ around the central time location $\boldsymbol{\tau}$. In the same way as the Fourier Transform, the Gabor Transform then converts the now time filtered function into Frequency Space analysing the computing the frequencies individually though out infinitesimal pieces of time. The reason that the Fourier Transform is insufficient in many cases such as the one we face in this project, is that it is time independent, meaning we lose all information about time in frequency space (See Hw1 - Submarine for more on Fourier Transform).

$$\text{Gabor Transform} : f(\tau, k) = \int_{-\infty}^{\infty} f(t)g(t-\tau)e^{-ikt}dt \tag{1}$$

$$\text{Inverse Gabor Transform} : f(\tau, k) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(t)g(t-\tau)e^{-ikt}dkd\tau \tag{2}$$

2.2 Shannon Filter

The Shannon Filter useful for its steep slope approaching 0 and wide top which allows for a clean filtering around a desired domain, in our case, allowing us to isolate a range of frequencies to pick out from an audio file with several frequency emitters present. This is Implemented through multiplying a function by a range (a through b) of allowed frequencies (k)

$$\text{Shannon Filter} : shannon\_filter = a < k < b \tag{3}$$

3 Algorithm Implementation and Development

For the implementation of the Gabor Transform algorithmicaly we once again refer to the Fourier Transform, this time looking at the Discrete Fourier Transform (See Hw1 - Submarine Section 3.1 for more detail). For a 14 second file in this case, we are centering around each point in time (t) with center tau. We do this in increments of 0.2 seconds here to get adequate results without overloading the software to the point where this loop crashes as there are expensive operations happening in the loop with large matrix multiplication. "a" determines the filter window size with an inverse relation to its size, so the larger "a" the smaller the window. We must keep in mind that the smaller the window the less information is being accounted for per iteration, but the larger the window, the less precise the result is to that point it time. This leads to a game of finding a window size that contains enough data without compromising to much precision. The algorithmic potion filters the signal in the time domain such that we are localized in time, transforms the signal to Frequency Space, and adds the frequency array to a new index in a matrix that at the end of the loop contains all frequency arrays in chronological order. This results in the ability to analyze what frequencies appear at each point in time.

Algorithm 1: Gabor Transform

```
tau = 0:0.2:14; % centre of window
a = 200; % window size
for j = 1:length(tau) do
    g_filter = exp(-a.*(t-tau(j)).^2); %Gaussian
    y_filtered = g_filter .* y_transpose;
y_filt_trans = fft(y_filtered);
Sgt_spec(:,j) = fftshift(abs(y_filt_trans));
end for
```

4 Computational Results

4.1 Music Scores

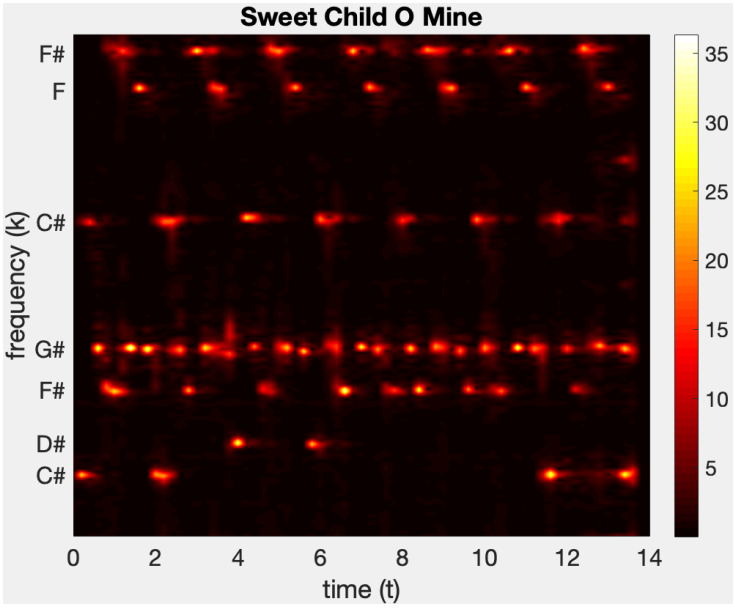

Figure 1: Music score of the 14 second audio file containing a portion of the song Sweet Chile O Mine.

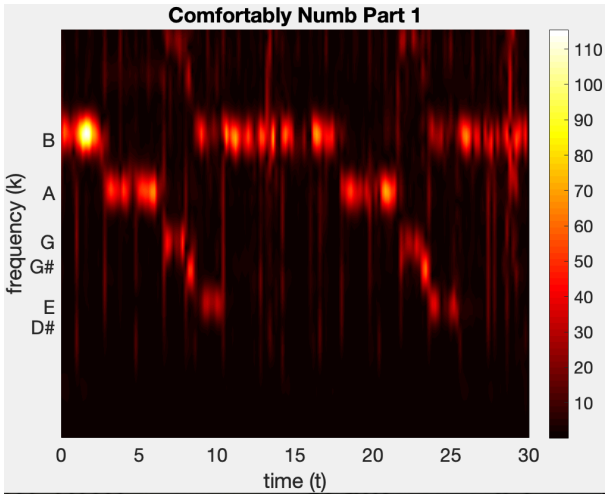

Figure 2: Comfortably Numb Part 1 Music Score

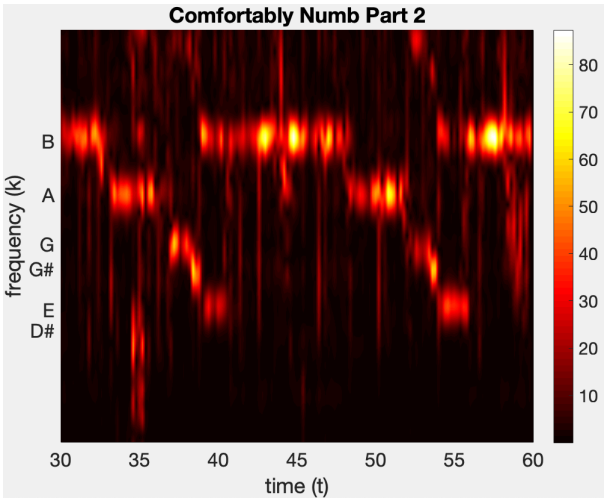

Figure 3: Comfortably Numb Part 2 Music Score

By plotting frequencies that appear at discrete points in time on a heat chart with a time in second x-axis and frequency in Hz y-axis, we visually represent the existence of the audio frequencies through time. Now we can label the y-axis with the notes associated with the frequencies at their respective spot, and show when each particular note is being played.

4.2 Isolated Instruments

Using figures 2 and 3 we can actually see the range of frequencies in which the bass is playing. This is because the notes are manually inserted along the y-axis so the numerical frequencies can be if seen if the y tick labels are removed in the source code. Figures 2 and 3 are close ups in the frequency range from 50 Hz to 150 Hz. Roughly speaking this is where the bass audio frequencies exist - the top edge of the range may go up to around 200 Hz and the bottom may go up to around 100 Hz. We can now filter the entire audio file in a single frame of frequency space around this frequency range, then convert it back to the time domain. This is because time has no significance in this part of the process, we are simply cleaning the entire file of a particular frequency range, and converting it back to the time domain where we retain all information on time. Now when we apply the Gabor Transform we can zoom out and see that other frequencies no longer exist in the array. We have effectively isolated the bass. We can now do the same thing when looking to isolate the guitar in Comfortably Numb, even though it is not nearly as prominent as the bass In Comfortably Numb or the guitar in Sweet Child O Mine. We do this by filtering out the frequencies we know don't encompass the guitar solo; aka, the frequencies that encompass the bass. By converting the filtered file back to the time domain we can plug it into the audio player, and listen to the instruments that are pick up in the allotted frequency range. We find that the guitar exists alone in frequency space roughly from 300 Hz to 4000 Hz with the most prominent frequencies around the 300 to 1000 Hz range. Below 300 mark it shares frequency space with snares, and 4000 it shares it with cymbals. The smaller we make this frequency range the more isolated the guitar becomes, but we lose more and more of the rich sound of the guitar as this range shrinks.

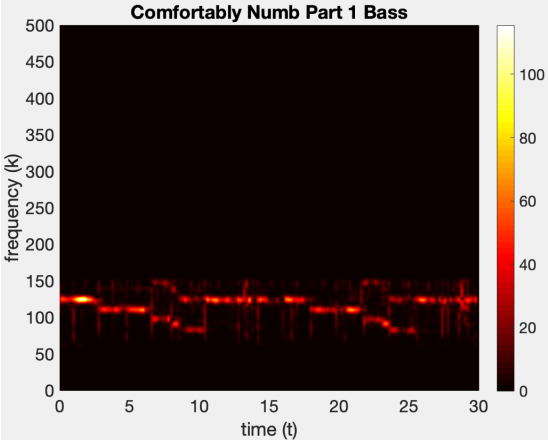

Figure 4: Comfortably Numb Bass Part 1

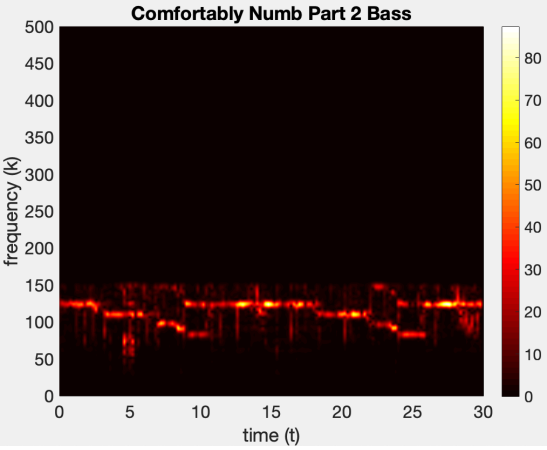

Figure 5: Comfortably Numb Part Bass 2

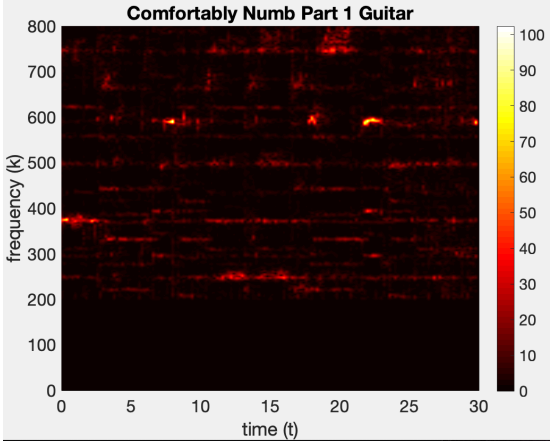

Figure 6: Comfortably Numb Guitar Part 1

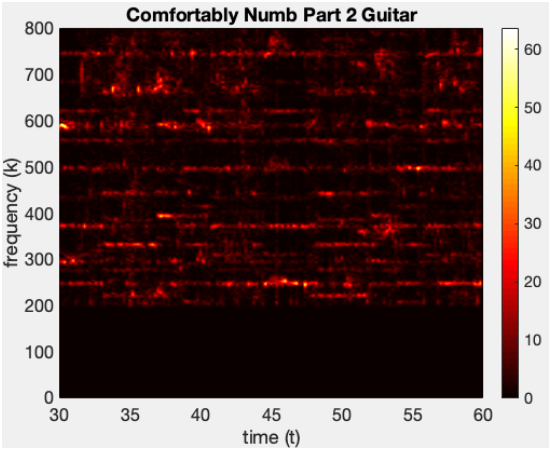

Figure 7: Comfortably Numb Guitar Part 2

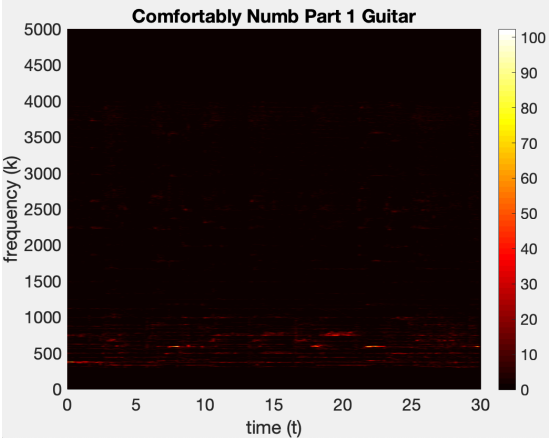

Figure 8: Comfortably Numb Guitar Part 1 Wide Range

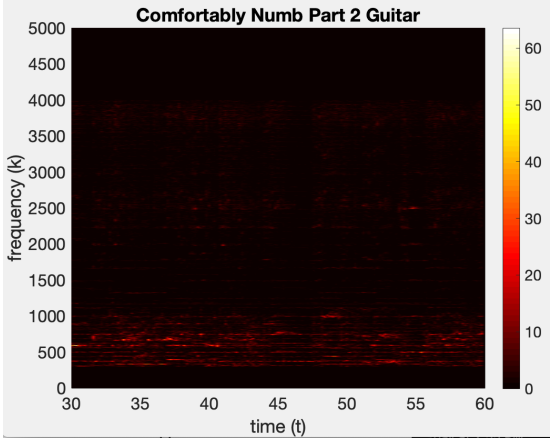

Figure 9: Comfortably Numb Guitar Part 2 Wide Range

5 Summary and Conclusions

Using the Gabor Transform it is Possible to isolate frequencies in time, and associate them with their respective positions in time. By doing this we can clearly identify the frequency range that is associated with a particular emitter and utilize this data to filter out or around desired ranges of frequency. This is exactly what we have been able to accomplish here, with the help of the Shannon Filter, to identify and isolating the bass frequency from an audio file containing the song comfortably numb, then isolating the guitar frequency. The end result will leave us with a new array that can now be plugged back into an audio player (here we use playblocking()) and listen to the isolated range.

Appendix A MATLAB Functions

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- Unt = fft(Un) returns the Fourier Transformed matrix Unt of array Un in any.
- Un = ifft(Unt) returns the inverse Fourier Transformed array Un of matrix Unt.
- fftshift(X) switches the left and right halves of the elements contained in X such that the 0 frequency component is centered along its axis
- [Y, Fs] = audioread(FILENAME) reads an audio file specified by the character vector or string scalar FILENAME, returning the sampled data in Y and the sample rate Fs, in Hertz.
- audioplayer(Y, Fs) creates an audioplayer object for signal Y, using sample rate Fs. A handle to the object is returned.
- `playblocking(OBJ)` plays from beginning; does not return until playback completes.
- `pcolor(C)` is a pseudocolor or "checkerboard" plot of matrix C. The values of the elements of C specify the color in each cell of the plot. In the default shading mode, 'faceted', each cell has a constant color and the last row and column of C are not used. With shading('interp'), each cell has color resulting from bilinear interpolation of the color at its four vertices and all elements of C are used. The smallest and largest elements of C are assigned the first and last colors given in the color table; colors for the remainder of the elements in C are determined by table-lookup within the remainder of the color table.
- `pcolor(X,Y,C)` where X and Y are vectors or matrices, makes a pseudocolor plot on the grid defined by X and Y. X and Y could define the grid for a "disk", for example.
- `colormap(MAP)` sets the current figure's colormap to MAP.
- `hot` Red-yellow-white color map inspired by black body radiation
- `colorbar` colorbar appends a colorbar to the current axes in the default (right) location
- `yticks(ticks)` specifies the values for the tick marks along the y-axis of the current axes. Specify ticks as a vector of increasing values, for example, [0 2 4 6].
- `yticklabels(labels)` specifies the y-axis tick labels for the current axes. Specify labels as a cell array of character vectors, for example, 'January','February','March'. This command will also set both YTickMode and YTickLabelMode to 'manual'.

Appendix B MATLAB Code

Double percent symbols (%%) are what separate sections in the source code, and are used to run each section individually and represent correlation of components of code. The code has also been divided by page accordingly to its relative relevance.

```
% Clean workspace
clear all; close all; clc

% In Order to run the program and extract calculations, this script must be
% run in sections such that only the data desired is loaded and only the
% computations desired are computed.

%%                                          % File Load In
%%
figure(1) % 14 seconds
[y, Fs] = audioread('GNR.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);

%%
figure(2) % 60 seconds
[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('Comfortably Numb');
p8 = audioplayer(y,Fs);
%playblocking(p8);
```

Listing 1:

```
%%                                          % Frequency and Time Domain song 1
%%
L = tr_gnr;   % time slot to transform
n = length(y); % number of Fourier modes 2^9
k = (1/L)*[0:n/2-1 -n/2:-1]; % Sweet child O Mine
k_shift = fftshift(k);
t2 = linspace(0,L,n+1);
t = t2(1:n);
y_transpose = y.';
%%                                          % Ploting Frequencies song 1

%       Gabor and Spect
tau = 0:0.2:14; % centre of window
a = 200; % window size

for j = 1:length(tau)
    g_filter = exp(-a.*(t-tau(j)).^2); % Gaussian
    y_filtered = g_filter .* y_transpose;
    y_filt_trans = fft(y_filtered);
    Sgt_spec(:,j) = fftshift(abs(y_filt_trans));
end

figure(3)
pcolor(tau,k_shift,Sgt_spec)
shading interp
set(gca,'ylim',[210 760],'Fontsize',16)
colormap(hot)
colorbar
xlabel('time (t)'), ylabel('frequency (k)')
yticks([277 311 370 415 554 698 739])
yticklabels({'C#', 'D#', 'F#', 'G#', 'C#', 'F', 'F#'})
title('Sweet Child O Mine');
```

Listing 2:

```
%%                                          % Frequency and Time Domain song 2
%%
y_transpose = y.';
%y_transpose = y_transpose(1:1317960); % only for Comfortably Numb part 1
y_transpose = y_transpose(1317961:1317960*2); % only for Comfortably Numb part 2
tr_gnr = length(y_transpose)/Fs;
L = tr_gnr;   % time slot to transform
n = length(y_transpose); % number of Fourier modes 2^9
k = (1/L)*[0:n/2-1 -n/2:-1]; % Comfort Numb
k_shift = fftshift(k);
t2 = linspace(0,L,n+1);
t = t2(1:n);
%%                                          % Ploting Frequencies song 2
tau = 0:0.2:30; % centre of window
a = 200; % window size

for j = 1:length(tau)
    g_filter = exp(-a.*(t-tau(j)).^2); % Gaussian
    y_filtered = g_filter .* y_transpose;
    y_filt_trans = fft(y_filtered);
    Sgt_spec(:,j) = fftshift(abs(y_filt_trans));
end

figure(3)
pcolor(tau,k_shift,Sgt_spec)
shading interp
set(gca,'ylim',[50 150],'Fontsize',16)
colormap(hot)
colorbar
xlabel('time (t)'), ylabel('frequency (k)')
title('Comfortably Numb Part 2');
yticks([77 82 92 97.99 110 123])
yticklabels({'D#','E','G#','G','A','B'})
xticks([0 5 10 15 20 25 30]) % part 2
xticklabels({'30','35','40','45','50','55','60'}) % part 2
```

Listing 3:

```
%%                                          % Frequency and Time Domain song 2 bass
%%
y_transpose = y.';
y_transpose = y_transpose(1:1317960); % only for Comfortably Numb
%y_transpose = y_transpose(1317961:1317960*2); % only for Comfortably Numb part 2
tr_gnr = length(y_transpose)/Fs;
L = tr_gnr;   % time slot to transform
n = length(y_transpose); % number of Fourier modes 2^9
k = (1/L)*[0:n/2-1 -n/2:-1]; % Sweet child O Mine
k = (1/L)*[0:n/2-1 -n/2:-1]; % Comfort Numb
k_shift = fftshift(k);
t2 = linspace(0,L,n+1);
t = t2(1:n);
y_transform = fft(y_transpose);
shannon_filter = abs(k_shift) < 200;
y_trans_filt = y_transform .* fftshift(shannon_filter);
y_inverse = ifft(y_trans_filt);

    % Play Full Range then Guitar Only
y_guitar = y_inverse.';
%Full
p8 = audioplayer(y,Fs);
playblocking(p8);
%Guitar
p8 = audioplayer(y_guitar,Fs);
playblocking(p8);
%%                                          % Ploting Frequencies song 2 bass
tau = 0:0.2:30; % centre of window
a = 200; % window size

for j = 1:length(tau)
    g_filter = exp(-a.*(t-tau(j)).^2); % Gaussian
    y_filtered = g_filter .* y_inverse;
    y_filt_trans = fft(y_filtered);
    Sgt_spec(:,j) = fftshift(abs(y_filt_trans));
end

figure(3)
pcolor(tau,k_shift,Sgt_spec)
shading interp
set(gca,'ylim',[0 500],'Fontsize',16)
colormap(hot)
colorbar
xlabel('time (t)'), ylabel('frequency (k)')
title('Comfortably Numb Part 1 Bass')
```

Listing 4:

```
%%                                          % Frequency and Time Domain song 2 Guitar
%%
y_transpose = y.';
y_transpose = y_transpose(1:1317960); % only for Comfortably Numb
%y_transpose = y_transpose(1317961:1317960*2); % only for Comfortably Numb part 2
tr_gnr = length(y_transpose)/Fs;
L = tr_gnr;   % time slot to transform
n = length(y_transpose); % number of Fourier modes 2^9
k = (1/L)*[0:n/2-1 -n/2:-1]; % Sweet child O Mine
k = (1/L)*[0:n/2-1 -n/2:-1]; % Comfort Numb
k_shift = fftshift(k);
t2 = linspace(0,L,n+1);
t = t2(1:n);
y_transform = fft(y_transpose);
shannon_filter = abs(k_shift) > 300;
shannon_filter1 = abs(k_shift) < 4000;
y_trans_filt = y_transform .* fftshift(shannon_filter).* fftshift(shannon_filter1);
y_inverse = ifft(y_trans_filt);

    % Play Full Range then Guitar Only
y_guitar = y_inverse.';
    %Full
%p8 = audioplayer(y,Fs);
%playblocking(p8);
    %Guitar
p8 = audioplayer(y_guitar,Fs);
playblocking(p8);
%%                                          % Ploting Frequencies song 2 Guitar
tau = 0:0.2:30; % centre of window
a = 200; % window size

for j = 1:length(tau)
    g_filter = exp(-a.*(t-tau(j)).^2); % Gaussian
    y_filtered = g_filter .* y_inverse;
    y_filt_trans = fft(y_filtered);
    Sgt_spec(:,j) = fftshift(abs(y_filt_trans));
end

figure(3)
pcolor(tau,k_shift,Sgt_spec)
shading interp
set(gca,'ylim',[0 5000],'Fontsize',16)
colormap(hot)
colorbar
xlabel('time (t)'), ylabel('frequency (k)')
title('Comfortably Numb Part 1 Guitar')
% xticks([0 5 10 15 20 25 30]) % part 2
% xticklabels({'30','35','40','45','50','55','60'}) % part 2
```

Listing 5: