AMATH 482 Homework 1

Zach Zlepper

February 5, 2021

Abstract

Given a song, how can sound engineers amplify certain instruments? How can they remove certain instruments? Over this paper we will go through a possible technique using Gabor Transforms to see what frequencies are being played at certain time steps. We will also use filtering to try and remove certain instruments to better see other less dominate instruments.

1 Introduction and Overview

We are given two songs, one Guns N' Roses song (GNR), Sweet Child O Mine, and a Pink Floyd song, Comfortably Numb. The GNR song is only a guitar solo without any other instruments. This allows us to use just a Gabor transform to isolate the guitar. However, with the Floyd song, we must use filtering to isolate the bass and multiple filters to better find the guitar.

2 Theoretical Background

We have seen that the Fourier transform decomposes a time signal into its frequency components, and is defined by

$$F(k) = \frac{1}{\sqrt{2}\pi} \int_{-\infty}^{\infty} e^{-ikx} f(x) dx \tag{1}$$

This inverse Fourier Transform is defined by

$$f(x) = \frac{1}{\sqrt{2}\pi} \int_{-\infty}^{\infty} e^{ikx} F(k) dk \tag{2}$$

A downside to the Fourier Transform is that in changing to the frequency domain knowledge about when frequencies occurred. We will use an algorithm called the Gabor Transform which allows us to take the Fourier Transform of sub domains to better isolate the time when frequencies were played.

$$\tilde{f}_g(\tau, k) = \int_{-\infty}^{\infty} f(t) g(t - \tau) e^{-ikt}\, dt \tag{3}$$

So, for a fixed τ we can see the frequency component around that time. The Gabor transform depends on the filter g.

- The function g must be real and symmetric.
- $||g||_2 := (\int_{-\infty}^{\infty} |g(t)|^2 dt)^2 = 1$, meaning the L-2 Norm of f is set to unity

There is also an Inverse Gabor Transform to which takes sub domains and glues their frequencies together. For both Algorithms, we can define our own filter as one described above. Specifically, the Gaussian filter satisfies those two requirements so we will pick the Gaussian for this paper.

Even with the sub domains, we still have to be conscious of the Heisenberg Uncertainty Principle. So the larger our Tau is the more frequency information we get. Due to the Heisenberg uncertainty principle, this also means we lose the ability to associate a specific time to the frequency.

Also, in reality we must take a discrete version of the Gabor Transform where $k = m * \omega_0$ and $\tau = n * t_0$. Our equation now becomes

$$\tilde{f}_g(\tau, m * \omega_0) = \int_{-\infty}^{\infty} f(t)g(t - n * t_0)e^{-im*\omega_0 t}\, dt \tag{4}$$

. Where ω_0, t_0 are positive constants defining the frequency space.

3 Algorithm Implementation and Development

For our algorithm, we are taking in a signal and outputting a vector with the frequency components at each time in order to plot a spectogram of the data. This is our algorithm for part 1 of the project.

Since we are wanting to be in hertz, we are scaling our k vector by $\frac{1}{L}$ instead of $\frac{2*pi}{L}$ during our set up.

Algorithm 1: Gabor Transform

Initialize variables and load song
for every time step τ **do**
  Build a Gaussian filter around τ with width a
  Filter signal in time domain by Gaussian
  Take the Fourier Transform
  Shift the Fourier domain and input frequencies at current τ into matrix with frequencies at other times
end for
Plot Spectogram

The a in the Gaussian filter determines the horizontal vs vertical focus of the Gabor Transform. A small a makes the Spectrogram highlight the frequencies more specifically but loses its time focus. This focuses horizontally. If we wanted a more vertical focus, a large a would give us more information about when the frequency was played but would limit the specificity of the frequency. To build a beautiful spectogram, we need to play around with a.

For part 1, we largely will use Algorithm 1 to plot a spectogram.

For part 2 of our assignment, we use Algorithm 1 but we added a filter around the frequencies that the bass plays before doing our Gabor transform. We can also take the maximum of each Gabor Transform which helps us visualize the frequencies attached to the Bass section.

Once we can identify the core bass frequency, we can filter out the harmonics from the frequency domain. We can do this using several adapted Gaussian filters that are focused on the frequencies we want to take out. These filters remove frequencies instead of keep frequencies so we need to change their formula appropriately. This better allows us to see the guitar and we can iterate further removing components that are either noise or harmonics.

For part 3 once we clean up our score by removing parts of the bass which were easily visible and their harmonics, we still had a lot of information left. I thought about averaging, but this isn't white noise that is remaining. It is the harmonics of other instruments and sounds. So averaging wouldn't work as it doesn't have a mean of 0.

4 Computational Results

See Figure 1 for a filtered version of Figure 4 around bass and a clearer picture in 2. We can see the bass around 125 hz and then its harmornics at 250 hz which are an A and a B respectively. We can pick up a nice Guitar score from the GNR clip with figure 3. With notes F, F, C, G. Ideally, we could pull this same clip for the Floyd Guitar but it is a difficult process since there are other instruments playing simultaneously. I tried to remove the bass harmonics to clean up the score. This was not easy and required constant iteration. Extra computing power would have helped. Here is an attempt in figure 5. For all the clips, I could only process the first few seconds of the song due to limited compute.

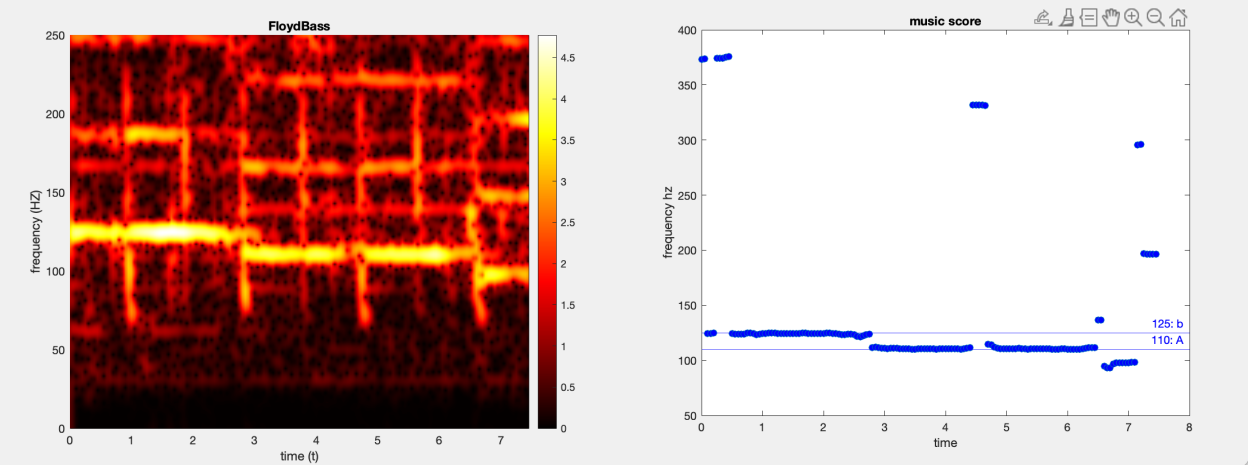

Figure 1: Bass line with a gaussian filter.

Figure 2: Bass line with filter and max pulled

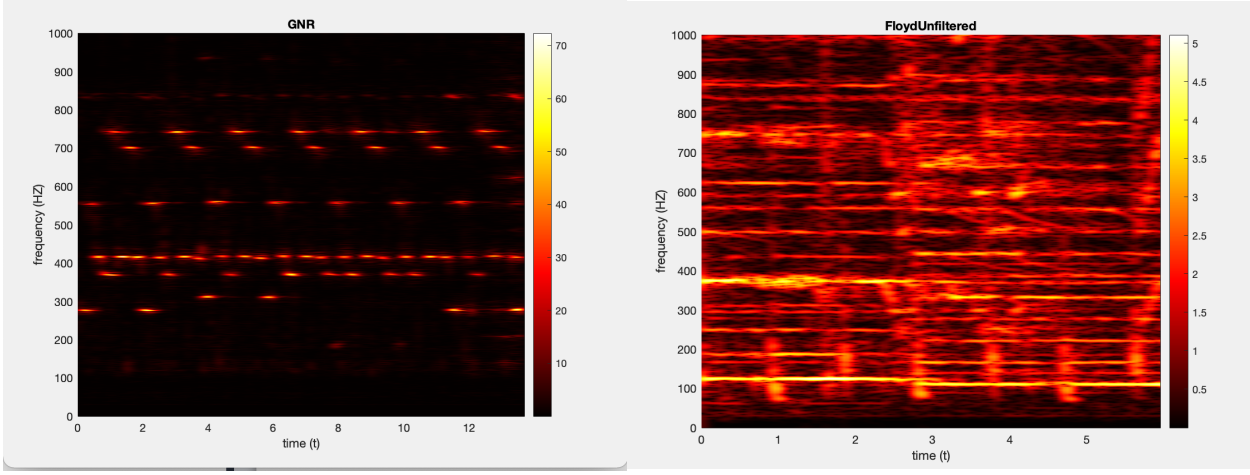

Figure 3: Here is the GNR song.

Figure 4: Here is the Floyd song unfiltered.

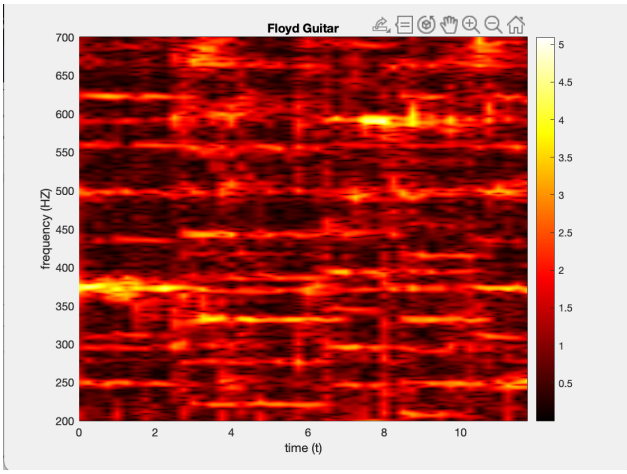

Figure 5: Here is the Floyd song with a few filters.

5 Summary and Conclusions

After using our Gabor Transform, we can clearly see the bass line in the Floyd song and the Guitar line in the GNR song. After filtering the Floyd song we can focus in on the bass line to pull out the bass frequencies. We show a possible theory on how to extract the Guitar line from the GNR song but as discussed, it is still hard to see due to limited compute and requirement of a lot of iterations.

I have realize that this is the first project which pushed my computer to its limits. It is actually interesting since computing power is something modern day computers do not have to worry about. I can only imagine how careful programmers had to be to insure there weren't any bugs in there code since it would take days to run in the past. Also, this project helped me appreciate runtime calculations and writing efficient code.

Appendix A

MATLAB functions used and implementation

audioread(file) : Takes in audio clip and returns a vector of amplitudes and sampling rate. abs(X) : Returns the absolute value of every element in X, or complex magnitude if the element is complex.

fftn(X) : Performs a N-D Fast Fourier Transform on X.

fftshift(X): Places the the zero-frequency components of X and the output of fftn to the center of the array.

ifftn(X) : Performs a N-D Inverse Fast Fourier Transform on X.

linspace(x1,x2,n) : It generates a linearly spaced vector with n evenly space points

$[X, Y, Z]$ = meshgrid(x,y,z) : Creates 3-D grid coordinates defined by the X, Y, and Z vectors.

max(A) : Returns the maximum value in vector A.

$[M, I]$ = max(A) : Returns the maximum value M and the index I of the matrix.

zeros(A,B,C) : Initializes matrix of size A, B, C with zeros.

Appendix A MATLAB Code

```
%% FLOYD BASS
[y, Fs] = audioread('Floyd.m4a');
trgnr = length(y)/Fs; % record time in seconds
% plot((1:length(y))/Fs,y);
% xlabel('Time [sec]'); ylabel('Amplitude');
% title('Comfortable Numb');
%p8 = audioplayer(y(1:length(y)/5),Fs); playblocking(p8);

%%

%rect = @(x, a, b) ones(1, numel(x)).*(a < abs(x)<b);


vec = y.';

L= trgnr;
n=length(y);
t2=linspace(0,L,n+1);
t=t2(1:n);

k=(1/L)*[0:n/2 -n/2:-1];ks=fftshift(k);

%filterBelow = rect(ks, 0, 250);
filter = exp(-3e-6*(ks).^2);
vec_ft = fft(vec);
vec_filter = vec_ft.*fftshift(filter);
y = ifft(vec_filter);

%%
a = 200;
tau = 0:.05:L/8;
notes = zeros(1, length(tau));
for j = 1:length(tau)
    g = exp(-a*(t-tau(j)).^2);
    Sg = g.*y;
    Sgt = fft(Sg);
    [M, I] = max(Sgt);
    notes(1, j) = abs(k(I));
    Sgt_spec(:,j) = fftshift(abs(Sgt));
end

%%

figure(4)
pcolor(tau, ks, Sgt_spec);
shading interp
colormap(hot)
colorbar
xlabel('time (t)'), ylabel('frequency (HZ)')
title('FloydBass 2nd ')
ylim([0,250])
%%

plot(tau, notes, 'o', 'MarkerFaceColor', 'b');
title('music score')
ylabel('frequency hz')
xlabel('time')
yline(110, 'b', '110: A')
yline(125, 'b', '125: b')
```