AMATH 582 Homework 1

Tomas Perez

January 27, 2020

Abstract

In this paper, a Gabor Transform with a Gaussian kernel is implemented to clean up audio tracks from Guns n' Roses' "Sweet Child O' Mine" and Pink Floyd's "Comfortably Numb". Using different filtering techniques with the Gaussian kernel we can clear up the noise in the track and distinguish individual notes from the songs played by the guitar and bass.

1 Introduction and Overview

Fourier Transforms are great at pulling the frequency content of a signal, but do not allow you to determine where those frequencies occur over a given time period. The Gabor Transform, also referred to as the Short-Time Fourier Transform (STFT), is a version of the Fourier Transform that is expanded to help examine the frequency content of a signal at various point in time. In the same way we use filtering with Fourier Transforms, we can introduce different filtering techniques to the Gabor Transform to help clean out excess noise. This makes it the ideal tool for analyzing audio signals from music. With this time-frequency technique we can determine the notes played by different instruments in "Sweet Child O' Mine" by Guns n' Roses and "Comfortably Numb" by Pink Floyd. Specifically we want to clean up the guitar from "Sweet Child O' Mine", and isolate the guitar and bass from "Comfortably Numb".

To visualize the frequency components as distinct notes from the songs, we can use spectrograms. A spectrogram is a visual representation of the frequency spectrum of a signal over time. In simplest terms, it's a picture of sound. Often displayed as a heat map, the intensity of a note will be a brighter color on the image.

As it stands, the data is noisy and in "Comfortably Numb" the bass and guitar are mixed within the sound track. The Gabor Transform accepts various filters, window sizes, and step intervals to analyze different points of a signal. In this paper, we will rely on the Gaussian filter. The Gaussian filter was discussed in the first homework and will not be explored in depth here. The

2 Theoretical Background

In this paper, we use the Gabor Transform (STFT) with a Gaussian kernel to examine frequency content of a signal in different windows of time. Fourier analysis tells us that any function $f(x)$ in the interval $x \in (-L, L]$ can be represented by a trigonometric series of sines and cosines. This greatly simplifies the evaluation of certain equations, including our evaluation of the acoustic signals from the submarine. Fourier Transform can be written as

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{\frac{in\pi}{L}x} \quad x \in (-L, L] \tag{1}$$

The Gabor Transform (Short-Time Fourier Transform) is defined as such,

$$G[f](t, w) = \int_{-\infty}^{\infty} f(\tau)\bar{g}(\tau - t)e^{-iw\tau}\, d\tau \tag{2}$$

where the filter function $g(\tau - t)$ is applied to $f(\tau)$ to localize the signal. The window of the the filter is centered at $t = \tau$ and everything outside of that window is zeroed. Sliding the filter window over the time domain of the signal allows the frequency content of the signal to be analyzed at each locality.

An important decision is the step interval and window size of the filter. Wide filter windows catch more frequency information, at the cost of precision. Narrow filters have great time resolution, but lose frequency information. Depending on the step interval and window size it is possible to oversample and undersample the data. Oversampling occurs when the windows at each step overlap with the window prior. Undersampling occurs when there is no overlap in the window at each step.

The filter function of choice for this paper is the simple Gaussian filer:

$$g_f(t) = e^{-\tau(t-t_0)^2} \tag{3}$$

3 Algorithm Implementation and Development

To isolate the notes of the songs load and construct the time domain of the signals and the wave numbers (k). In this analysis we will rescale the wave numbers by $2\pi/L$. The conversion the frequency measured in Hertz is done with the following relationship:

$$w = \frac{2\pi}{L} = 2\pi f \tag{4}$$

Then set the step interval and window size for the filter. Looping over the step interval, we construct the Gaussian filter at each point and multiply it with the signal. Transform the signal (FFT) and save the absolute value of the shifted transformed signal in to a matrix. We use this to construct the spectrogram with time as the x-axis and frequency in Hz as the y-axis.

Algorithm 1: Gabor Loop

```
Import data from audio.m4a
Create time and frequency domains
Set window size ($\tau$) and step interval ($tslide$)
for $j = 1 : length(tslide)$ do
  Construct Gaussian filter $g(t)$
  Multiply the signal and filter
  Take the FFT of the filtered signal
  Take the absolute value and shift the transformed signal
  Save to a $j$ row matrix
end for
Build spectrogram
```

When examining ”Comfortably Numb” we will want to further clean the data with a Gaussian filter in the frequency domain. This requires a small addition to the algorithm.

Algorithm 2: Gabor Loop with Filter in Frequency Domain

```
Import data from audio.m4a
Create time and frequency domains
Set window size ($\tau$) and step interval ($tslide$)
for $j = 1 : length(tslide)$ do
    Construct Gaussian filter in time domain $g(t)$
    Multiply the signal and filter
    Take the FFT of the filtered signal (Sgt)
    Take the absolute value and shift the transformed signal
    Take the max values
    Use the indices to construct a Gaussian filter in frequency domain $g(k)$
    Shift $Sgt$ and multply by the new filter
    Save the absolute value to a $j$ row matrix
end for
Build spectrogram
```

4 Computational Results

4.1 Gabor Tranform with Gaussian Filter in Time Domain

Figure 1 shows the frequency in Hz of the notes played in the guitar solo from "Sweet Child O' Mine" by Guns n' Roses. Figure 2 shows the frequency in Hz of the notes played by the bass during the guitar solo in "Comfortably Numb" by Pink Floyd. Note that the y-axis is shortened for the spectrogram of "Comfortably Numb". The y-axes of the spectrograms were chosen based on the frequency range of a 6-string guitar and a 4-string bass (the kind of bass used in "Comfortably Numb"). A standard 6-string guitar has a frequency range roughly between 80Hz and 1100Hz, while a 4-string bass has a range of roughly 40Hz to 320Hz.

Figure 1: Sweet Child O' Mine

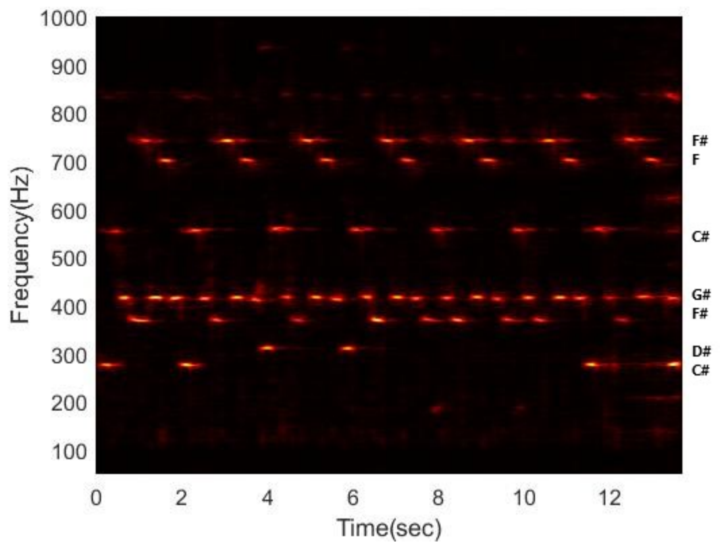
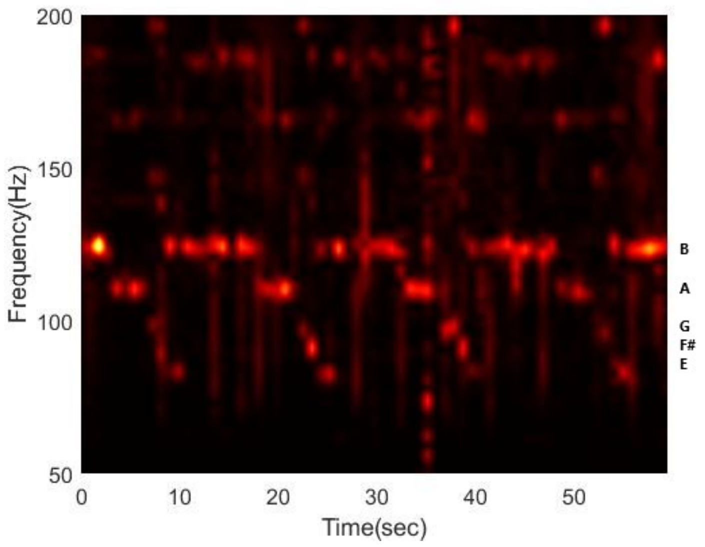


Figure 2: Comfortably Numb (Bass)



"Comfortably Numb" presents a separate challenge. The bass and guitar both appear on the spectrogram. In Figure 2, the guitar is mostly cut out, but the frequencies overlap at a certain point. The lowest note played by the guitar is a C# note on the low E-string, which has a frequency of approximately 116Hz. Conversely, the highest note played by the bass is a B note played on the G-string, which has a frequency of approximately 123Hz. This will be touched on more in the break out of "Comfortably Numb".

4.2 "Comfortably Numb": Adding a Gaussian Filter in the Frequency Domain

Figure 3 shows the frequencies of the notes played by the bass. Figure 4 shows the frequencies of the notes played by the guitar. The spectrograms are much cleaner with the notes being more defined after the addition of a second filter in the frequency domain. Practically all noise has been removed. In Figure 3, the guitar note that blends into the bass frequencies is visible. Unfortunately, it looks like some of the notes were lost with the additional filter, possibly due to undersampling.

5 Summary and Conclusions

This study shows the usefulness of Gabor Transforms in signal processing. Music is essentially varying frequencies over time. Implementing this time-frequency analysis technique allowed us to get a clear image of the notes being played by the guitar and bass instruments, and in doing so we can essentially reconstruct the score of the songs. Adding a filter in the frequency domain provided a remarkably clear spectrogram. However, it is possible that in doing so some of the frequency information was lost from undersampling.

Figure 3: Comfortably Numb (Bass)

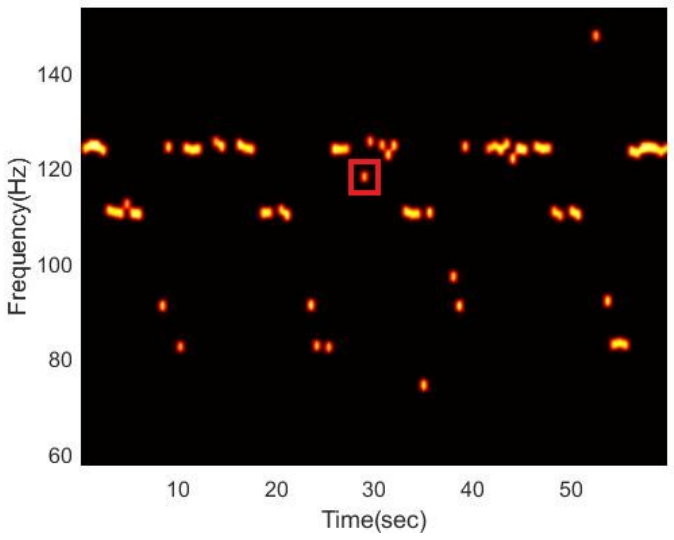
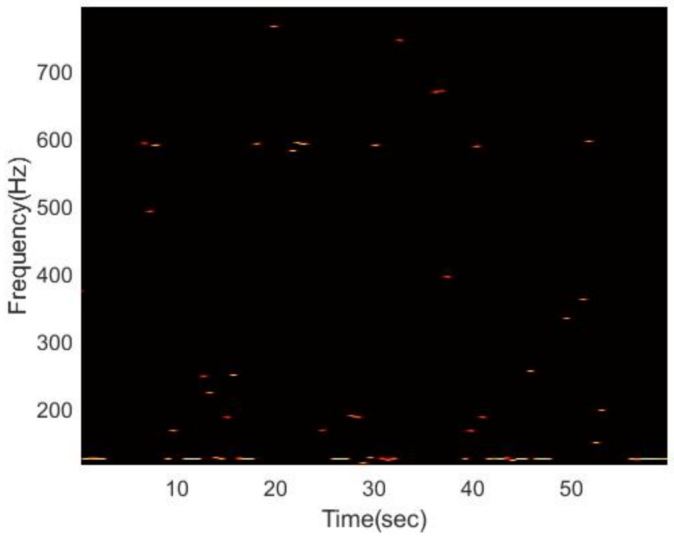


Figure 4: Comfortably Numb (Guitar)



Appendix A MATLAB Functions

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- `ks = fftshift(k)` rearranges a Fourier transform `X` by shifting the zero-frequency component to the center of the array.
- `U = fft(X)` computes the DFT of X using a FFT algorithm.
- `[row,col] = ind2sub(V,I)` returns the arrays row and col containing the equivalent row and column subscripts corresponding to the linear indices `I` for a matrix of size `V`.
- `[M,I] = max(A)` returns the maximum element of an array and the index into the operating dimension that corresponds to the maximum value of `A`.
- `pcolor(X,Y,X)` specifies the X and Y vertices and creates a pseudocolor plot using the values of matrix C.

Appendix B MATLAB Code

```
%% GNR
close all; clear all; clc;
[y,Fs] = audioread('GNR.m4a');
L=length(y)/Fs; n=length(y);
S=y.';
t2=linspace(0,L,n+1); t=t2(1:n);
k=(2*pi/L)*[0:n/2-1 -n/2:-1]; ks=fftshift(k);
tau=100; tslide=0:0.1:L;
sgt_spec=zeros(length(tslide),n);
for j=1:length(tslide)
    g=exp(-tau*(t-tslide(j)).^2);
    sg=g.*S;
    sgt=fft(sg);
    sgt_spec(j,:)=abs(fftshift(sgt));
end
pcolor(tslide, ks/(2*pi), sgt_spec'), shading interp
set(gca,'Ylim',[50 1000],'Fontsize',[12]) %guitar range
xlabel('Time(sec)'); ylabel('Frequency(Hz)'); colormap(hot)
%% Pink Floyd
close all; clear all; clc;
[y2,Fs] = audioread('Floyd.m4a');
y=y2(1:end-1); S=y.';
L=length(y)/Fs; n=length(y);
t2=linspace(0,L,n+1); t=t2(1:n);
k=(2*pi/L)*[0:n/2-1 -n/2:-1]; ks=fftshift(k);
tau=100; tslide=0:0.9:L;
sgt_spec=zeros(length(tslide),n);
for j=1:length(tslide)
    g=exp(-tau*(t-tslide(j)).^2);
    sg=g.*S;
    sgt=fft(sg);
    sgt_spec(j,:)=abs(fftshift(sgt));
end
pcolor(tslide, ks/(2*pi), sgt_spec.'), shading interp
set(gca,'Ylim',[50 200],'Fontsize',[12]); %4string bass range
xlabel('Time(sec)'); ylabel('Frequency(Hz)'); colormap(hot);
%% Floyd, freq filter
close all; clear all; clc;
[y2,Fs]=audioread('Floyd.m4a');
y=y2(1:end-1); S=y.';
L=length(y)/Fs; n=length(y);
t2=linspace(0,L,n+1); t=t2(1:n);
k=(2*pi/L)*[0:n/2-1 -n/2:-1]; ks=fftshift(k);
tau=50; tslide=linspace(0,t(end),100);
sgt_spec=zeros(length(tslide),n);
for j=1:length(tslide)
    g=exp(-tau*(t-tslide(j)).^2);
    sgt=fft(g.*S);
    Sgts=abs(fftshift(sgt));
    [M,I]=max(Sgts(n/2:end));
    [I1,I2]=ind2sub(size(Sgts),I+n/2-1);
    g2=exp(-0.5*((ks-ks(I2)).^2));
    Sgts2=fftshift(sgt).*g2;
    sgt_spec(j,:)=abs(Sgts2);
end
pcolor(tslide, ks/(2*pi), log(sgt_spec.'+1)), shading interp
set(gca,'Ylim',[0 1000],'Fontsize',[12])
xlabel('Time(sec)'); ylabel('Frequency(Hz)'); colormap(hot)
```

Listing 1: Gabor Transform Code

```
%%  play just the bass using butter filter
close all; clear all; clc;
[y2, Fs] = audioread('Floyd.m4a');
fc=180; %freq cutoff
fn=Fs/2; %nyquist freq
[b,a]=butter(5,fc/fn);
freqz(b,a)
SF=filter(b,a,y2);
p8=audioplayer(SF,Fs);
playblocking(p8);
```

Listing 2: BONUS: Play just the bass with a Butter filter applied (I used the butter command Dr. Kutz!!!