AMATH 582 Homework 2: Gabor Transform and Spectrogram for Audio Analysis

Hongda Li

February 6, 2021

Abstract

In this paper, we are interested in applying the principal of Gabor filtering on real world audio data. Our objective is to utilize different filtering techniques using wavelets to extract out sounds for certain instrument in the audio data. In addition, we also discuss the best way of choosing parameters for Gabor filtering to get the best visualizations for the spectrogram. The results we present are musical notes of the guitar solo and bass solo for 2 rock songs.

1 Introduction and Overview

The objective is to reproduce the musical scores for a clip taken from the start of the song "*Sweet Child O' Mine*" by Guns N' Roses and a guitar solo from ''Comfortably Numb" by Pink Floyd. We deploy the technique of first using a Shannon Filter on the whole audio to filter out only the frequencies for the instruments that we are interested in, using common knowledge and some basics in music, and then we use Gabor transform to find the best temporal and frequencies resolutions. Finally, we present a way to visualize the spectrogram, and an algorithm to identify the notes with some basics in music theory. In addition, we also explored ways of truncating the audio data so that it runs more efficiently.

2 Theoretical Background

The Gabor transform consider the additional usage of a kernel function in the time frequencies, preferably a function that has bounded L2 Norm over the real complex domain. And this is discussed more in details in [1], we will summarize the important piece here.

$$g_{\tau,w}(\tau) = \exp(i\omega\tau)g(\tau - t) \tag{1}$$

ω is the multiplier for the Fourier Kernel. g is a bounded function for filtering signal near t. A specific case of the Gabor transform is called the Short Time Fourier Transform(STFT), where we consider a real symmetric matrix to be the kernel. Together with the function $f(\tau)$ representing the signal, we have:

$$\widetilde{f}_g(t, w) = \int_{-\infty}^{\infty} f(\tau)g(\tau - t)\exp(-i\omega t)d\tau \tag{2}$$

Notice that, the variance under the time and frequencies domain are bounded by some constant. This means that, shortening the width of the kernel will reduce Frequency resolution while increasing the with will reduce time resolution. The problem can be addressed with some oversampling to produce better visualization to compensate. In addition, we consider the usage of several wavelet function for filtering in the time and frequency domain of the signal.

$$g(\tau; t, w) := \exp\left(-\left(\frac{\tau - t}{\frac{\sqrt{2}w}{2}}\right)^2\right) \tag{wavelet.G}$$

$$g(\tau; t, w) := \begin{cases} 1 & |\tau - t| \le w/2 \\ 0 & |\tau - t| > w/2 \end{cases} \tag{wavelet.S}$$

Where each of the wavelet function (Will be referred to as filter in the text) has one time parameter τ as the input, and its location and with is controlled by the parameters: t, w. In the case of the Gaussian Filter, w means a distribution with $2w$ standard deviation. For computational purposes, the filtering processing using wavelets are implemented as element-wise vector multiplications and the STFT is implemented via FFT with a for loop that slide the kernel through the domain.

3 Algorithm Implementation and Development

Two core parts of the algorithm is the parameters for the STFT, the implementations of STFT and the conversion between time domain vector and the frequencies domain vector, other aspects of the algorithm will be discussed as we view the results. The whole procedure of the processing any given audio can be summarized by:

1. Setting up the parameters for the spectrogram, choosing a section of the music to analyze.
2. FFT on the data, filter out a range of frequencies that the instrument is plying in with the Shannon Filter.
3. Create the spectrogram, truncate it to get the best view for the musical scores, and find out the peaking frequencies.

3.1 Time to Frequencies Domain Conversion

The time domain vector is by a vector with the same length as the number of floats in the audio data. As common knowledge, digital audio data usually are sampled 480000 times second (Uncompressed m4a format).

Algorithm 1: Converting Time Domain to Frequencies Domain

```
Input: TimeVec
n := length(TimeVec); L := TimeVec(end) - TimeVec(1);
if n is even then
    hz:=$\frac{2\pi}{L}[0 : n/2 - 1, -n/2 : -1]$
else
    hz:=$\frac{2\pi}{L}[0, 1 : (n - 1)/2, -(n - 1)/2 : -1]$
end if
hz:= $\frac{\text{hz}}{2\pi}$
Output:  FFTshift(hz)
```

Notice that, depending on the parity of the width of the signal, we need to partition the corresponding frequencies domain differently. In the case of even number of partitions, we want 0 appears at index $n/2 - 1$ after the fftshift, assuming index start with 0, and we want 0 to be at index $\lfloor n/2 \rfloor$ after fftshift assuming index starts with 0. Most importantly, we need to divide it by 2π to actually obtain the frequencies for the audio.

3.2 Creation of spectrogram Using STFT

The next part is the creation of the spectrogram. We implement the width of the filter to be relative to the window's size, and the window's size is simply the total length divides by the number of partitions we want. This is convenient when we sync up the STFT with the bars of music, which helps us get better results and visualization.

Algorithm 2: Creating spectrogram with STFT

```
1:  Input: TimeVec: Tvec, Audio: A, RelWidth: W, Number of Partitions: N, WaveletFunction: g
2:  Initialize: SpectroMatrix
3:  dt := (Tvec(end) - Tvec(1))/n
4:  Hz := Input Tvec into Algorithm 1
5:  for II = 0:N - 1 do
6:      t := Tvec(0) + II*dt
7:      F := g(Tvec; Tvec(1) + II*dt + dt/2, W*dt)
8:      Signal := F*A
9:      Signal := fftshift(fft(Signal))
10:     Signal := Filter out excessive frequencies using Hz vector.
11:     SpecMatrix(:, II + 1) = Signal
12:     SpecMatrix(:, II + 1) = Normalize the II + 1 th row of SpecMatrix, and put it into log scale
13: end for
14: Output: Tvec, Hz, SpecMatrix
```

This algorithm (Algorithm 2) is generic for all audio and wavelet function: A, and g. The variable "RelWidth:W" represents the with of the wavelet relative to the with of the partition of the signal in time domain, this adds more flexibility for simple oversampling, in addition, if "N × W": is set to be a constant, then the amount of oversampling is kept unchanged

In addition, extra processing (line 12 in Algorithm 2) is involved to make better presentation on the spectrogram, trimming of frequencies that is negative and outside the range of the instrument we are interested in, and present it on a lot scale to scale down the relative sizes of the frequencies constant making the spectrogram more invisible.

3.3 Global Frequencies Filtering and Audio Truncations

By common knowledge, music are written in bars of notes in Chromatic Scale or the Diatonic Scales for pop songs (Diatonic Scale is in the Chromatic Scale). Each note in Chromatic Scale increments by a multiplier of $2^{1/12}$ in term of frequencies (A semi-tone). An instrument such as guitar usually operates inside a range of 3 octaves (12 Semi-tone per octave).

The reasoning of filtering out the frequencies for just the instrument makes us easier to identify particular instruments using the spectrogram. Implementations wise this is achieved via applying a Shannon Filter to the FFT of an audio section we are interested in, and then Inverse Transform it back. The process acts as a band pass filter and it isolate one of the instruments from other instruments.

Audio is truncated either by seconds, or by BPM (Beats per minute), both are implemented. The reasoning is that, music repeats around a theme with some variations, and they are located in sections of bars. Truncating the music in terms of bars reduce loads for computations, it also makes better visualization for the spectrogram.

4 Computational Results

4.1 GNR: Notes and Spectrogram

For the clip from GNR (*Sweet Child O' Mine*), a shannon filter located at the interval $[220, 800]$ is applied, it covers the alto an tenor range of the music, the expected range for the guitar. There are 8 bars of music in total, each 2 bars repeats and builds on the same motif, the first 3 sections build on variations, while the 4th one repeats the first 2 bars.

After running the routine highted in the previous section, the result is shown in figure 1. The melody is written on the C# major key. The music score is: C#3,C#4, G#3, F#3,F#4, G#3, F4, G#3, the 2 bars is then repeated with variations on the first, second and 4th keys.

Computationally, The first bars of music is partitioned into 128 sections, and the Gabor kernel is the Gaussian function with a relative width of 4, this allows the audio to be super-sampled, allowing for an smoother image. Other wavelets for STFT is experimented however Gaussian and Shannon filter was proven to produce the best spectrogram and the most accurate results.

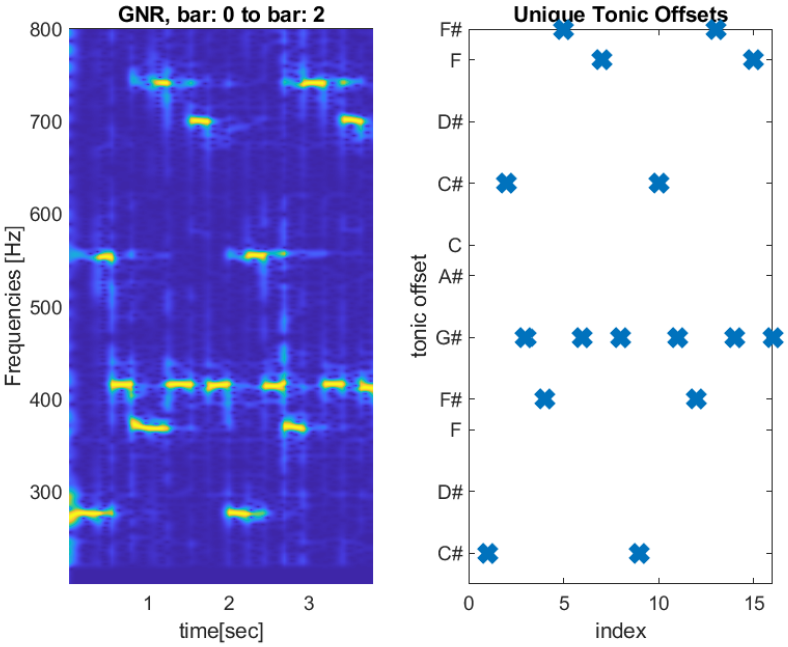

Figure 1: GNR: Spectrogram of the Guitar, Gaussian Wavelet

4.2 Floyd: Bass Notes and Spectrogram

The bass of the song “*Comfortably Numb*” performs a Basso Obstinate on the G minor scale. A Shannon Filter is used globally to filter out frequencies in the $[50, 110]$ hertz Because that is the expected acoustic range off the instrument. The result is presented in Figure 2. It shows one bars of the score that is played byt the bass which is repeated through out the audio clip. The music score is: D1, B1, A1, G1, F1, D1. However, it’s unclear whether the key B, and D are presented at the same time to create a harmony.

For this example, both the Gaussian and the Shannon Wavelet function is deployed. Take note that Figure 2 Left shows the spectrogram when the Guassian Wavelet is used and Figure 2 Right shows the spectrogram when Shannon Wavelet is used. Both Clearly identifies the same frequencies content, however it visually Shannon Wavelet produces sharper image.

The audio clip is partitioned into 64 sections. (Observe how much bigger it is compare to audio clip); and a relative with of 4 is chosen. The choice is justified by the fact that lower acoustic signal need longer window to identify.

4.3 Floyd: Guitar Solo and Spectrogram

In this section, we will present the spectrogram for the guitar solo in the audio clip. However, due to the virtuosity of the guitar soloist, the music score cannot be presented. More sophisticated algorithm is need to identify complex harmony and melodic contour, unfortunate it’s not as simple as STFT.

The routine is run on the whole music clip for Floyd’s “Comfortably Numb” on the acoustic range of the guitar via global filtering, and the results in shown in: Figure 3. Please observe the complexity of the melodic contour, and how simple the previous 2 examples are. In this case, multiple notes seems to be played at the same time. The notes are Legato instead of staccato, creating significant challenge for reading the

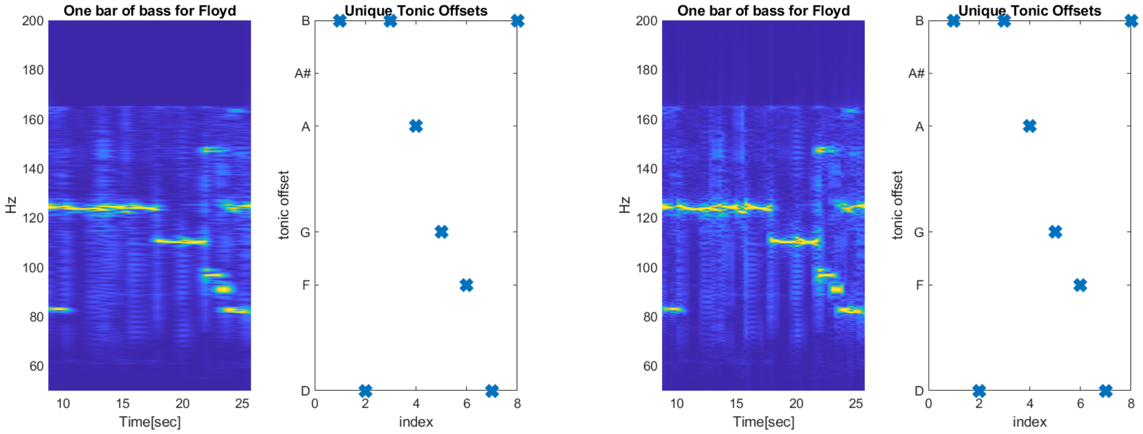

Figure 2: Floyd's Bass, Gaussian and Shannon Wavelet

notes from the spectrogram. Simply trick by pulling out the frequencies with the maximal magnitude is not feasible anymore.

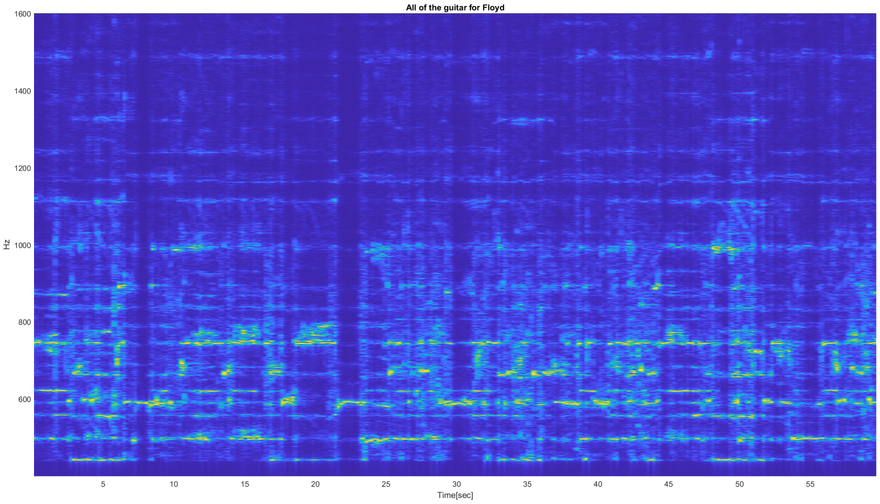

Figure 3: Floyd whole Song Guitar with Shannon Wavelet

5 Summary and Conclusions

The algorithmic frameworks for the STFT can be easily implemented in high level scientific programming language. The visualization process is however, where mostly the works comes in. To get the best out of the digital signal, we need domain specific knowledge for analysis, trial errors to figure out the best window with and discretization of the signal. However, it also shows us that, when dealing with more complex melodic contour and musical articulations, significant challenge is placed on the process of interpreting the spectrogram as music score.

References

[1] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013, p. 324.

Appendix A MATLAB Functions