AMATH 482 Homework 2

Seoyoung Cho

February 10, 2021

Abstract

Although the Fourier Transform is a powerful technique to analyze signals in frequency space, it has one drawback. After shifting and taking the absolute of the Fourier Transformed signals, all the time information about when each frequency occurs is lost. In this assignment, we take account of this limitation and introduce a new technique called the Gabor Transform. With the Gabor Transform, we can obtain an optimal balance of information in both frequency and time domains. Using this technique, we reproduce the music scores of guitars and bass in two rock songs "Sweet Child O' Mine" and "Comfortably Numb".

1 Introduction and Overview

"Sweet Child O' Mine" by Guns N Roses and "Comfortably Numb" by Pink Floyd are two well-known rock music of all time. We will refer to each as "GNR" and "Floyd" respectively for the sake of brevity. Given short audio clips of both pieces of music, the task given is to reproduce music scores of the guitar in "Sweet Child O' Mine" and that of bass in "Comfortably Numb" using the Gabor filtering. With the given audio clips, we can import and convert the music into vector form which allows modification of music data. Then we would apply the concept of the Gabor Transform in the time domain to perform the task. Further, for the task of reproducing the music score of "Comfortably Numb", we are asked to perform an additional step of isolating the bass and the guitar separately. In order to isolate only the specified instruments, we filter the signal data in the frequency domain. To achieve the task, we introduce a new type of filter called, the Shannon filter. This paper will discuss both the theoretical backgrounds of the techniques used and the implementation process to achieve the tasks. Figure 4 in Appendix C shows the plots of the given audio data of the two songs.

2 Theoretical Background

This section explains the mathematical concepts that are crucial in approaching the task.

2.1 The Gabor Transform

The concept of the Gabor Transform stems from the question: "How can we obtain both time and frequency information in the signal analysis?". As mentioned earlier, the Fourier Transform is a powerful tool that allows analysis of frequency from a signal but has one critical drawback of losing the time information as it integrates out time to calculate the frequency. In other words, the Fourier Transform can tell the exact frequency of a signal but not when that frequency is emitted. To complement the problem, the Gabor Transform applies the Fourier Transform to signals in each time increment. This is achieved by applying the filter while sliding the time window along with the domain. The Gabor transform is a mathematical function that is multiplied by the signal data in the time domain. Given τ center time, $a > 0$ width of the window, the filter function for the Gabor Transform is the following:

$$\tilde{f}_g(\tau, k) = \int_{-\infty}^{\infty} f(t)g(t-\tau)e^{-ikt}\,dt \tag{1}$$

Basically, we integrate over the product of the original signal and the filter function centered at τ. The result depends on the choice of the filter function and they require specific properties. The filters for the Gabor Transform should be: (i) real and symmetric (ii) L_2 norm of the function is 1. Although the latter can be any constant, it makes the inverse Gabor Transform less complicated. For simplicity, we use the Gaussian equation as our filter function:

$$e^{-a(t-\tau)^2} \tag{2}$$

The inverse Gabor Transform which recovers the signal data in the time domain is the following:

$$f(t) = \frac{1}{2\pi||g||_2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \tilde{f}_g(\tau, k) g(t-\tau) e^{-ikt} \, dk d\tau \tag{3}$$

The width of the filter window plays an important role in obtaining information in both frequency and time domains. If the window is too large, the Gabor Transform works the same as the Fourier Transform and delivers only the frequency data. On the other hand, if the window is too small, then we will be considering the individual time increment without much frequency data. Hence, we need to decide the optimal width value that balances the amount of time and frequency information. Now with the Gabor Transform, we can analyze when each frequency is detected. However, there is still a problem. Just like the Fourier Transform, the Gabor Transform takes a continuous signal function, unlike the given audio data that is finite and discrete in time. Therefore, we will use the Discrete Gabor Transform.

2.2 Discrete Gabor Transform

The Discrete Gabor Transform takes in the discrete and finite signal data. Since the data is discrete, we cannot slide the time window arbitrarily. Instead, we take account of the discrete set of both frequencies k and center time τ. Given $m, n \in \mathbb{Z}$ and positive frequency resolutions ω_0, t_0,

$$k = m\omega_0$$

$$\tau = nt_0$$

The Discrete Gabor Transform is the following:

$$\tilde{f}_g(m, n) = \int_{-\infty}^{\infty} f(t) g(t - nt_0) e^{2\pi i m \omega_0 t} \, dt \tag{4}$$

Having small enough t_0 and large enough ω_0 will optimize the amount of time and frequency data. Using the Gaussian equation, below is our Gabor filter $g(t-\tau)$:

$$g(t-\tau) = e^{-a(t-\tau)^2} \tag{5}$$

To apply the filter, we multiply the above equation to the Fourier Transformed signal data over the time domain. The Figure 5 in Appendix C captures the process of the Gabor Transform being applied to "Sweet Child O' Mine" and "Comfortably Numb". The peaks showing up in the first plots for both songs will slide throughout the whole time domain.

2.3 Spectrogram

Through the Gabor Transform, we obtain both frequency and time data. Referring back to Figure 5, we can see that the third plots, which are simply the Fourier Transform visualizations, only show the frequency data. Without the first and the second plots, the time information of each frequency can't be deciphered. Hence, to convey the obtained frequency and time data, we use a spectrogram. The spectrogram is a visual representation of the strength of frequencies of signal data as it varies over time τ. The spectrogram of our Gabor transformed data will be helpful in the way it displays the frequency corresponding to the time information. Considering the music data, the spectrogram is the representation of each music note being played at that moment. Therefore, we will reproduce the music scores of "Sweet Child O' Mine" and "Comfortably Numb" using the spectrograms. Examples of the spectrograms will be delivered in Section 4.

2.4 Shannon filter

One important tool used to isolate instruments in Floyd's song is the Shannon filter. Shannon filter is a step function that is applied to the Fourier Transformed data in frequency space. Basically, the filter takes in the range of frequency values and attenuates all the data outside of the range. Hence, we use the Shannon filter to isolate the bass and the guitar in "Comfortably Numb". After inspecting the song, we decide the range of frequencies of the bass and the guitar separately. Then we can apply the Shannon filter to the specified ranges and reproduce the music scores of the isolated instruments. Figure **??** [3] in Appendix C shows the plot of the Shannon filter being applied in the frequency domain. The purple plot defines the range of frequency in which the data (in green) outside of the range are set to zero.

2.5 Guassian filter

When the music is played with guitar or bass, the plucking of the strings creates overtones. Overtones are the harmonics that are produced by the vibration of the strings. If the overtones are kept, we will see the fainter replica of the melody above the original notes being played on our spectrogram. Hence, filtering the overtones is necessary to clear out the spectrogram. We can achieve this with the Gaussian filter. The Gaussian filter works the same as the Gabor filter, multiplying the filter equation to data. One difference is that the Gaussian filter is applied in the frequency domain and not in the time domain. Given the central frequency (k_0) and the width of the filter (b), the Gaussian filter $F(k)$ is:

$$F(k) = e^{-b(k-k_0)^2} \tag{6}$$

Although the filter can be in different forms, we again use a Gaussian filter for simplicity. We can infer that the actual note being played will have the strongest intensity, and set such frequency as the center frequency k_0. Then applying the Gaussian filter around the k_0 will remove unnecessary frequencies, the overtones.

3 Algorithm Implementation and Development

This section is delivered in two parts according to each task for GNR and Floyd. Due to technical difficulties upon implementing the tasks for the Floyd clip, we truncated the song to 15 seconds and repeat the process for the first 30 seconds of the song. For more details of MATLAB functions, refer to Appendix A.

3.1 Initial setup

This initial setup applies to both GNR and Floyd songs.

- Load the audio clip of each song. This gives the amplitude vector and its rate of the song. The amplitude vector is an audio signal data in time-space.
- Define the time domain, frequency, and frequency grid vector. It is important to not multiply the frequency by 2π, because the data we are trying to get is the frequency in Hertz. Multiplying by 2π gives the angular frequency instead.
- Define τ, the time vector which will slide through the whole time domain. Each element of this vector is the center time where the Gabor Transform window will be centered at.
- Define the width a. This value decides the scope of the localization in the frequency window. Again we choose the a value that optimizes the balance of cleanliness of data in both time and frequency window.

3.2 Applying the Shannon Filter

This part is not necessary for GNR song which only plays guitar. For Floyd, multiple instruments are being played synchronously. Hence, we need to filter only the bass or the guitar to reproduce the music scores of both. Again, this is done by using the Shannon filter. The basic approach is as the following:

- Take the Fourier Transform of the amplitude vector. This will take the audio signal in the time domain to the frequency domain.
- Define a filter, a vector that contains all the indices of frequency that are not in the frequency range that the instrument plays. In Floyd, the bass doesn't go beyond the frequency of 200. This can be found looking at the music sheet ([5]-GNR [1][2]-Floyd) and referring to the frequency of each notes using the Note-Frequency chart [4]. The guitar has a more dynamic range of frequency at different portions of the song, hence we only focus on the first 10 seconds of the song. During that portion, the guitar solo ranges from the frequency of 500 to 1200 roughly. Define filter vector accordingly.
- Once the filter is defined, set values of the Fourier Transformed data at specified indices to be zero. This only gives the Fourier Transformed data within the frequency range of the instrument.
- Apply inverse Fourier Transform to the modified data. This will recover the audio signal data in a time-space that only contains the specified instrument.

For Floyd, the inverse Fourier Transformed data will be our new audio data which we will apply the Gabor Transform on. For GNR, we continue with the original audio data.

3.3 Applying the Gabor Transform

- For each time sliding increment τ, define a Gabor filter equation that centers at τ.
- Multiply the Gabor equation to the audio data. This process attenuates all the frequencies outside of the window. Again for GNR, we use the original data whereas, for Floyd, we use the Shannon filtered data. Note that the process is the same as in Gaussian filtering in the frequency domain, but only done in time-space.
- Take the Fourier Transform of the filtered data. This takes the audio data in the frequency domain.
- To get a cleaner spectrogram, we filter out the overtones. Again, for each slice of frequency data of each time slide, the actual notes being played by the instrument have the largest intensity. Hence, the index where the maximum of the absolute value of the Fourier Transformed data is at specifies the frequency of the actual note. Then we remove the overtones in each frequency slice by applying the Gaussian filter centered at the maximum frequency to the Fourier Transformed data. This process will clear out the overtones in each slice of frequency data.
- After, we apply the `fftshift` to the filtered data (both Gabor and Gaussian filter). This rearranges the Fourier Transformed data so we have zero-frequency data at the center. Then we take the absolute value of shifted data and store it as the spectrogram data. This data stores the energy of each frequency.

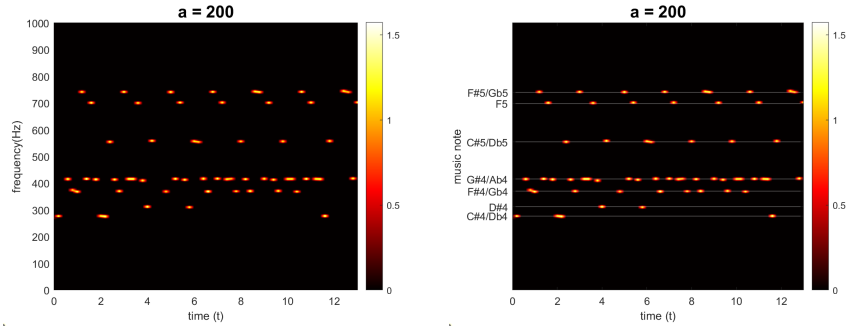
Now we can plot the spectrogram using `pcolor`. The spectrogram plots all the notes being played by the specified instrument. Then we compare each frequency of the notes to reproduce the actual music score.

4 Computational Results

This section discusses the music scores derived after implementation and development in MATLAB.

4.1 Music Score for GNR guitar

Figure 3 shows the spectrograms of the guitar playing in "Sweet Child O' Mine". Upon multiple trials, we decided the window width a of 200 optimizes the balance in the information of frequency and time. Notice in Figure 3a that each brighter spot in the plot shows the frequency corresponding to each note being played at the moment. Hence, we use the Note-Frequency Chart [[4]] to convert each frequency to music notes. This is shown in Figure 3b. We can simply read off the notes of the bass from the figure: "Db4 Db5 Ab4 F#4 F#5 Ab4 F Ab4". This pattern of melody continues throughout the end of the clip.



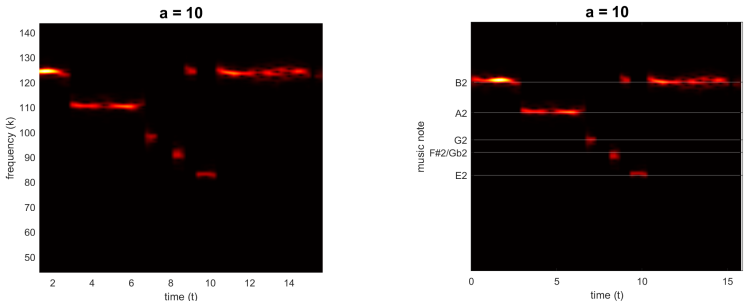
(a) Frequency vs Time Spectrogram (b) Music Note vs Time Spectrogram

Figure 1: Reproduced Music Scores of GNR

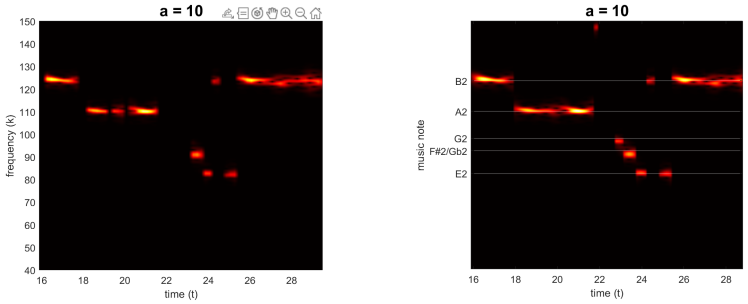
4.2 Music Score for Folyd bass and guitar

4.2.1 Floyd bass

Recall that unlike the GNR clip that only contains guitar melody, the Floyd clip contained melodies of multiple instruments: guitar, bass, drum, etc. Among all the instruments, the bass had the greatest intensity which made filtering of bass easier. Again, due to technical difficulties, Figure 2 shows the spectrograms of both frequency (Figure 2a, 2c) and music note (Figure 2b, 2d) overtime for the first half of the clip. The observed pattern of the bass repeats throughout the whole clip: "B2 A2 G2 F#2 E2 B2". Figure **??** in Appendix C shows the modified plot of the actual audio clip with only the bass included.

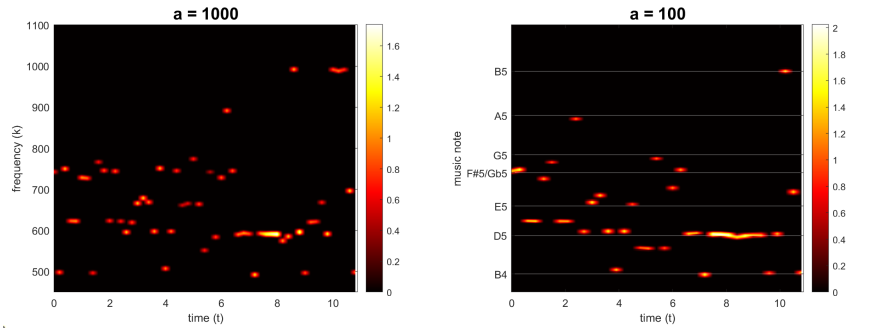


(a) Frequency vs Time Spectrogram pt 1 (b) Music Note vs Time Spectrogram pt 1



(c) Frequency vs Time Spectrogram pt 2 (d) Music Note vs Time Spectrogram pt 2

Figure 2: Reproduced Music score of Floyd



(a) Frequency vs Time Spectrogram (b) Music Note vs Time Spectrogram

Figure 3: Reproduced Music Scores of Floyd guitar solo

4.2.2 Floyd guitar

Unlike the bass that has a solid frequency range of up to 200Hz, the guitar has a dynamic range of frequency throughout the clip. Hence, we focus on the first 10 seconds, where frequency ranged from 500 to 1100. Apart from its dynamic frequency range, another problem was the overlapping of the drum notes jamming the solo melody. Since the bass has the most intensity, we first filter out the range of the bass. Then we isolate the maximum frequencies after applying the Gabor Transform. Figure **??** shows the resulting spectrogram after attempting to isolate the guitar solo. The guitar solo for the first 10 seconds should have the melody of: "E5 E5 D5 B4 D5 E5 E5 D5 B4 B4 D5 F#5 A5 B5" according to the guitar music sheet [2]. However, we can see that the plot is more scattered and only some spots lie on the melody line. We conjecture that this is due to the overlapping of the drum. Filtering the bass overtone and drum might enhance the result.

5 Summary and Conclusions

In this paper, we discussed the performance tactics to reproduce the music scores of 2 well-known rock music: "Sweet Child O' Mine" and "Comfortably Numb". Using the Gabor transform, the limitation of the Fourier Transform was resolved and was able to obtain when each frequency or note of the song is played. Thus, we were able to use the spectrogram to visualize the music score being played at each moment. Further, the effects of the Shannon and Gaussian filter in isolating one instrument and clearing overtones proved to be successful when the instrument emits the largest frequency within solid range. As the Fourier Transform, the Gabor Transform is another powerful technique to perform signal analysis.

References

[1] M. (2017, December 15). Comfortably Numb - Bass Cover Score and Tab [Video]. Retrieved from https://www.youtube.com/watch?v=NTMZLQ5b5Okamp;feature=youtu.be

[2] M. (2019, February 13). Sweet Child O Mine Guns N Roses Piano Tutorial Synthesia [Video]. Retrieved 2021, from https://www.youtube.com/watch?v=NswTA9vbqW4

[3] N. (Director). (2011, February 25). Comfortably Numb pulse solo TAB [Video file]. Retrieved 2021, from https://www.youtube.com/watch?v=f2fGJ7_uXmE

[4] Nyquist–Shannon sampling theorem. (2021, January 22). Retrieved February 10, 2021, from https://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem

[5] Tuning. (n.d.). Retrieved February 10, 2021, from https://pages.mtu.edu/ suits/notefreqs.html

Appendix A MATLAB Functions

This section lists all the important MATLAB Functions used in the process of submarine tracking.

- `[y,Fs] = audioread('file')`: reads the audio file and returns `y` the sampled data and `Fs` the sample rate for `y`.
- `linspace(x1,x2,n)`: returns a row vector of evenly spaced `n` points from `x1` to `x2`. This is used when creating time grid vector.
- `pcolor(X,Y,C)` creates a pseudocolor plot of the values in matrix C corresponding to the x and y coordinates. This is used to create spectrogram. `X,Y,C` are τ, frequency ks, and the Gabor transformed data respectively.
- `abs(X)`: returns the absolute value of each element in array `X`.
- `max(X)`: returns the maximum value in array `X`.
- `exp(x)`: returns the value of e^x.
- `log10(x)`: returns the vlaue of $logx$. Used to rescale the frequency intensity to get clearer sectrogram.
- `fft(A)`: returns the Discrete Fourier Transformation of signal data `A`. This uses the Fast Fourier Transform algorithm. Performing the DFT in MATLAB gives the Fourier values in the order of $\{\hat{x}_0, \hat{x}_1, \cdots, \hat{x}_{\frac{n}{2}}, \hat{x}_{\frac{-n}{2}+1}, \hat{x}_{\frac{-n}{2}+2}, \cdots, \hat{x}_{-1}\}$.
- `fftshift(A)`: returns the rearranged Fast Fourier transform with zero-frequency component shifted to the center.
- `ifft(At)`: returns the inverse Discrete Fourier transformation of the frequency data `At`. This uses the Fast Fourier Transform algorithm and returns the signal in real space.
- `plot(X,Y)`: Plots each coordinate given by `X,Y` in 2D space.
- `yticks(ticks)` sets the y-axis tick values. Use this to specify the music notes in spectrogram.
- `yticklabels(labels)` newly labels the y-axis tick values with the given label values. Use this to label the music notes in spectrogram.

Appendix B MATLAB Code

```
clear; close all; clc;

%% part 1: GNR

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

L = tr_gnr; n = length(y_g);
t2 = linspace(0,L,n+1); t = t2(1:n);
k = (1/L)*[0:n/2-1 -n/2:-1];ks = fftshift(k); %scaling by 1/L to get unit Hz instead of angular freq

%% Gabor transform

tau = 0:0.2:13;
a = 200;
gnr_spec = [];
for j = 1:length(tau)
    gabor = exp(-a*(t - tau(j)).^2);
    yg_g = gabor.*y_g;
    ygt_g = fft(yg_g);
    % filtering overtones
    [maxfreq(j), maxfreqind(j)] = max(abs(ygt_g));
    ygtf_g = ygt_g.*exp(-0.05*(k-k(maxfreqind(j))).^2);
    gnr_spec = [gnr_spec; abs(fftshift(ygtf_g))];
end

%% plotting example gabor transform

gabor = exp(-a*(t - 4).^2);
yg_g = gabor.*y_g;
figure(2)
subplot(3,1,1)
plot(t,y_g), hold on
plot(t,gabor, 'r');
xlabel('time (t)'), ylabel('y_g(t)')
subplot(3,1,2)
plot(t,yg_g)
xlabel('time (t)'), ylabel('y_g(t)*g(t-\tau)')
subplot(3,1,3)
plot(ks,abs(fftshift(fft(yg_g)))/max(abs(fft(yg_g))))
xlabel('frequency (k)'), ylabel('fft(y(t)*g(t-\tau))')

%% GNR spectrogram
```

```
figure(3)
pcolor(tau,ks,log10(abs(gnr_spec.')+1)), shading interp
colormap(hot)
axis([0 13 0 1000]);
% musical note
yticks([277.18 312 370 415.3 554.36 698.46 739.99]);
yticklabels({'C#4/Db4', 'D#4', 'F#4/Gb4', 'G#4/Ab4', 'C#5/Db5', 'F5' 'F#5/Gb5'});
colorbar;
ax = gca
ax.YGrid = 'on'
ax.Layer = 'top'
ax.GridAlpha = 0.5;
ax.GridColor = 'w';
%xlabel('time (t)'), ylabel('frequency (k)')
xlabel('time (t)'), ylabel('music note')
title(['a = ',num2str(a)],'Fontsize',16)

%% part 2: Floyd bass (first 15 seconds)
clear; close all; clc;

figure(3) % floyd audio plot
[y_f, Fs_f] = audioread('Floyd.m4a'); % y = sampled data; Fs = sample rate for y
y_f = y_f.';
 % record time in seconds
plot((1:length(y_f))/Fs_f,y_f);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Comfortably Numb');
%p8 = audioplayer(y_f,Fs_f); playblocking(p8);

% Gabor transform
y_f = y_f(1:700000);
tr_floyd = length(y_f)/Fs_f;
L = tr_floyd; n = length(y_f);
t2 = linspace(0,L,n+1); t = t2(1:n);
k = (1/L)*[0:n/2-1 -n/2:-1];ks = fftshift(k);
a = 10;
tau = 0:0.2:L;
floyd_spec = [];

%% (next 15 seconds)
clear; close all; clc;
[y_f, Fs_f] = audioread('Floyd.m4a'); % y = sampled data; Fs = sample rate for y
y_f = y_f.';
y_f = y_f(700001:1300000);
tr_floyd = length(y_f)/Fs_f; % record time in seconds
plot((700001:700000+length(y_f))/Fs_f,y_f);

% Gabor transform
L = tr_floyd; n = length(y_f);
t2 = linspace((700001/Fs_f),(700001/Fs_f)+L,n+1); t = t2(1:n);
k = (1/L)*[0:n/2-1 -n/2:-1];ks = fftshift(k);
a = 10;
tau = (700001/Fs_f):0.1:(700001/Fs_f)+L;
floyd_spec = [];
```

```
%% filtering bass using Shannon filter

yt_f = fft(y_f);
shannon = abs(k) >= 200;
yt_f(shannon) = 0;
yif_f = ifft(yt_f);

%% Gabor transform

for j = 1:length(tau)
    gabor = exp(-a*(t - tau(j)).^2);
    yg_f = gabor.*yif_f;
    ygt_f = fft(yg_f);
    % filtering overtones
    [bassfreq, bassfreqind(j)] = max(abs(ygt_f));
    ygtf_f = ygt_f.*exp(-0.05*(k-k(bassfreqind(j))).^2);
    floyd_spec = [floyd_spec; abs(fftshift(ygtf_f))];
end
%% sample diagram of gabor transform
gabor = exp(-a*(t - 5).^2);
yg_f = gabor.*y_f;

subplot(3,1,1)
plot(t,y_f), hold on
plot(t,gabor, 'r');
xlabel('time (t)'), ylabel('y_f(t)')
subplot(3,1,2)
plot(t,yg_f)
xlabel('time (t)'), ylabel('y_f(t)*g(t-\tau)')
subplot(3,1,3)
plot(ks,abs(fftshift(fft(yg_f)))/max(abs(fft(yg_f))))
xlabel('frequency (k)'), ylabel('fft(y(t)*g(t-\tau))')

%% Floyd bass spectrogram
% need to change the axis according to each part
figure(4)
pcolor(tau,ks,floyd_spec.'), shading interp
colormap(hot)
%axis([0 15.9 40 150]);
axis([15.9 28.8 40 150]); % part 2 axis
%music note
yticks([82.41 92.5 98 110 123.47]);
yticklabels({'E2', 'F#2/Gb2', 'G2', 'A2', 'B2'});
colorbar
ax = gca
ax.YGrid = 'on'
ax.Layer = 'top'
ax.GridAlpha = 0.5;
ax.GridColor = 'w';
%xlabel('time (t)'), ylabel('frequency (k)')
xlabel('time (t)'), ylabel('music note')
title(['a = ',num2str(a)],'Fontsize',16)
```

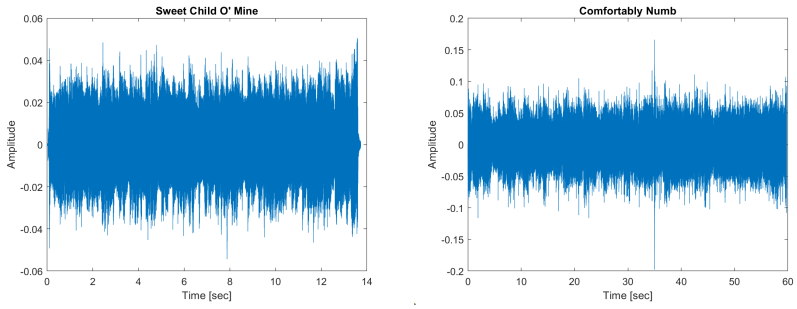
```
%%
plot(t, yif_f)
xlabel('Time [sec]'); ylabel('Amplitude');
title('Comfortably Numb bass');
p8 = audioplayer(yif_f,Fs_f); playblocking(p8);
audiowrite('floydbass.m4a',yif_f,Fs_f);
%% part 3: Floyd guitar (first 10 seconds)
clear; close all; clc;
% filtering guitar using Shannon filter
[y_f, Fs_f] = audioread('Floyd.m4a'); % y = sampled data; Fs = sample rate for y
y_f = y_f.';
y_f = y_f(1:480000);
tr_floyd = length(y_f)/Fs_f; % record time in seconds
%p8 = audioplayer(y_f,Fs_f); playblocking(p8);
% Gabor transform
L = tr_floyd; n = length(y_f);
t2 = linspace(0,L,n+1); t = t2(1:n);
k = (1/L)*[0:n/2-1 -n/2:-1];ks = fftshift(k);
a = 100;
tau = 0:0.3:L;
guitar_spec = [];

yt_f = fft(y_f);

% shannon filter
guitar1 = k <= 450;
guitar2 = k >= 1000;
yt_f(guitar1) = 0; yt_f(guitar2) = 0;
yif_f = ifft(yt_f);
%% Gabor transform
for j = 1:length(tau)
    gabor = exp(-a*(t - tau(j)).^2);
    yg_f = gabor.*yif_f;
    ygt_f = fft(yg_f);
    % filtering overtones
    [guitarfreq, guitarfreqind(j)] = max(abs(ygt_f));
    ygtf_f = ygt_f.*exp(-0.05*(k-k(guitarfreqind(j))).^2);
    guitar_spec = [guitar_spec; abs(fftshift(ygtf_f))];
end
%% Guitar Spectrogram
pcolor(tau,ks,log10(abs(guitar_spec.')+1)), shading interp
colormap(hot)
axis([0 L 450 1100]);
% music note
yticks([493.88 587.33 659.25 739.99 783.99 880 987.77]);
yticklabels({'B4','D5', 'E5' 'F#5/Gb5', 'G5', 'A5', 'B5'});
colorbar
ax = gca
ax.YGrid = 'on'
ax.Layer = 'top'
ax.GridAlpha = 0.5;
ax.GridColor = 'w';
%xlabel('time (t)'), ylabel('frequency (k)')
xlabel('time (t)'), ylabel('music note')
```

```
title(['a = ',num2str(a)],'Fontsize',16)
```

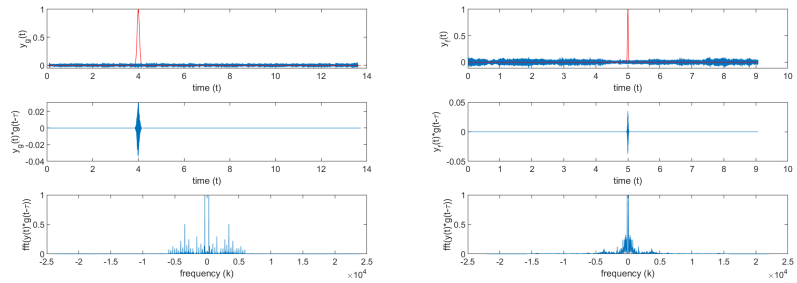
Appendix C Additional Figures



(a) Plot of "Sweet Child O' Mine" audio data

(b) Plot of "Comfortably Numb" audio data

Figure 4: Amplitude vs. Time plots



(a) The Gabor Transform applied to "Sweet child O' Mine"

(b) The Gabor Transform applied to "Comfortably Numb"

Figure 5: Snapshots of the Gabor Transform being applied

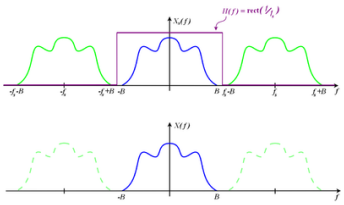


Figure 6: Example of Shannon filter

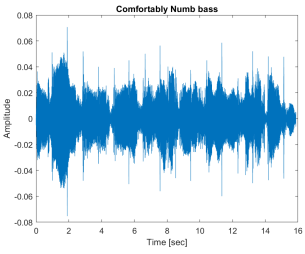


Figure 7: Plot of "Comfortably Numb" bass audio data