Ruimin Zhang
Feb 7th, 2020

Short-time Fourier Transform of Note Analysis

Abstract:

This is the homework 2 of AMATH 482 in the University of Washington. This assignment focuses on short-time Fourier transform, which provides us with a spectrogram of both time and frequency resolutions.

Sec. I. Introduction and Overview

This paper contains two parts – the analysis of Gábor transformation through scaling and translation parameters and application of two music waves. In part 1, we play with Gabor transformation a little bit to explore its behavior. In part 2, we apply the filter window to two pieces of music. If the width and translation constants are appropriate, we will be able to see both time and frequency resolutions and know when the notes are played and in what order (i.e., the correlation between time and frequency). Finally, we will examine the difference of overtone in the two music waves. Hopefully, we are able to unravel the neatness of short-time Fourier transform and see why it is useful.

Sec. II. Theoretical Background

2.1 Short-time Fourier transform(STFT)

The Gábor transform, also known as Short-time Fourier transform is defined as

$$\mathcal{G}(t,\omega) = \int_{-\infty}^{\infty} f(\tau)\overline{g}(\tau - t)e^{-i\omega\tau}d\tau \quad (1)$$

where $\overline{g}$ indicates the complex conjugate of the function.

To be more precise about the mathematical features, some assumptions about commonly used $g_{t,\omega}$ is considered. For convenience, we assume ① g to be real and symmetric; ② $\|g(t)\|_2 = 1$ where $\|g(t)\|_2 = \int_{-\infty}^{\infty}|g(t)|^2 dt$. Thus, the our modified STFT becomes

$$\mathcal{G}[f](t,\omega) = \int_{-\infty}^{\infty} f(\tau)g(\tau - t)e^{-i\omega\tau}d\tau \quad (2)$$

Since STFT is computed by discretizing the time and frequency domain, a discrete version of transform should be considered. Consider the sample points $v = m\omega_0$ and $\tau = nt_0$ where $m, n \in \mathbb{Z}$ and $\omega_0, t_0 > 0$ are constants. Then the discrete version of STFT is

$$\tilde{f}(m,n) = \int_{-\infty}^{\infty} f(t)g_{m,n}(t)dt \quad (3)$$

Then if $0 < \omega_0, t_0 < 1$, then the signal is over-sampled and time frames which yield excellent localization of the signal in both time and frequency. If $1 < \omega_0, t_0$, the signal is under-sampled and STFT lattice is unable to reproduce the signal. The overlap of the STFT window frames ensures good time and frequency resolutions of a given signal can be achieved.

2.2 Drawback of STFT

While STFT can give us information in both time and frequency domain, there are some limitations of STFT. Since the window is centered at τ with width a, any signal with wavelength longer than the window cannot be captured. If the window is too thin, then we lose wavelength information; if the window is too wide, then we lose time information -- an extreme would be Fourier transform where the width is $-\infty$ to ∞. Thus, choosing the appropriate τ and a is very important.

2.3 Common filter

1. Gaussian filter: $g = e^{-a(k-k_0)^2}$ where a is the width of the filter and k_0 is the center of the filter.

2. Mexican hat wavelet: $\psi(t) = (1-t^2)e^{\frac{-t^2}{2}}$. This wavelet is essentially a second moment of a Gaussian in the frequency domain. In our case, we use $\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a})$ where $a \neq 0$ and b are real constants. The parameter a is the scaling parameter that determines the width and b is the translation parameter that determines the center of the window, then we can shift the filter window by b and scale it by a.

3. Step-function(Shannon) window:

$$\psi(t) := \begin{cases} 1, & if\ |t| < \frac{1}{2} \\ 0 & otherwise \end{cases} \quad (4)$$

In our case, we use

$$\psi_{a,b}(t) := \begin{cases} a, & if\ |t-b| < \frac{1}{2} \\ 0 & otherwise \end{cases} \quad (5)$$

where a changes the width and b changes the center of the width.

Sec. III. Algorithm Implementation and Development

To do Gabor transform, we proceed three steps:

(i) Define the sampling frequency Fs, the number of points n, and amount of time L. Then define scaling parameter a and the translation parameter Δt. Then each filter will be centered at $t_0 + n\Delta t$ with width a. Define filter function.

(ii) Gabor transform. Take the FFT of the original function, then apply the filter to it.

(iii) Plot the spectrogram of the transformed result.

Part 1:

We want to analyze the same signal with different filters – Gaussian filter, Mexican hat wavelet, and step-function(Shannon) wavelet -- and see how the change in scaling parameter and translation parameter changes our resulting spectrogram.

Note:

1. Since I used the angular frequency at first, each time I plot in the frequency domain, I need to divide 2π from k in order to get the radial frequency. Then we have frequency in Hz.
2. In part 1, the number of points is not a multiple of 2, so we define k to be `(2*pi/L)*[0:(n-1)/2 -(n-1)/2:-1].`

Part 2:

We use Gabor transform to sample out the central frequencies. Then we are able to see which frequency is played first, which is played second, etc. If we map these frequencies back to the music scale figure for piano, we can find which note is played. Similarly for the recorder.

The reason why we use Gabor transform for a piece of music played on an instrument is that instrument causes overtone which is related to the timbre. The timbre is related to the overtones generated by the instrument for a center frequency. Thus if you are playing a note at frequency ω_0, an instrument will generate overtones at $2\omega_0$, $3\omega_0$, $\cdots$ and so forth. If we use

FFT, we cannot distinguish weather a frequency at a given point is overtone or central frequency. So we need information in both time and frequency domain. In this case, Gabor transform is our best option.

Sec. IV. Computational Results

4.1 Part 1

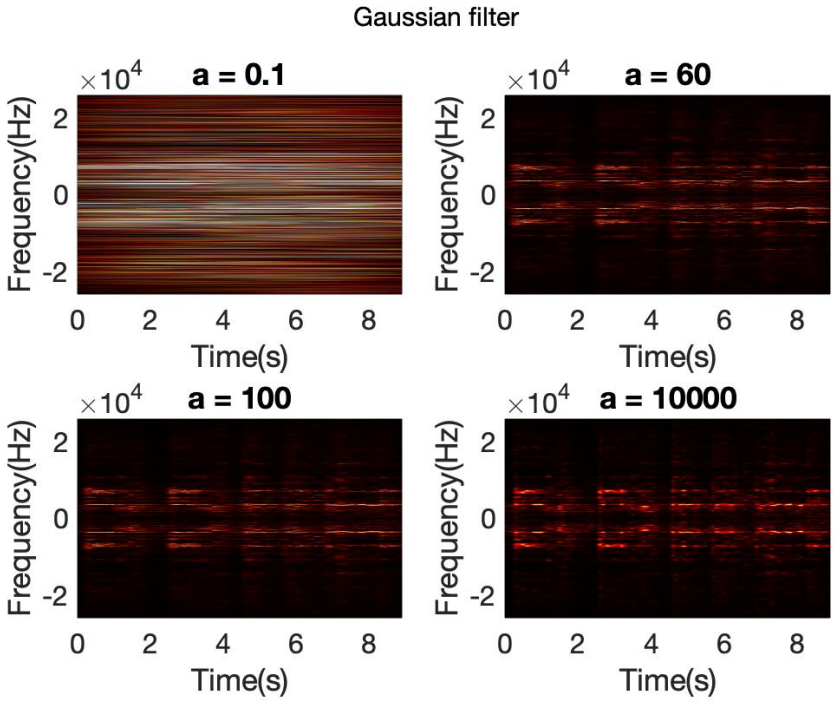

Figure 1. The spectrogram after we apply the gaussian filter. We can see that as a increases, the width of the filter decreases, resulting less frequency resolution and more time resolutions; as a decreases, the width of the filter increases, resulting more frequency but less time. At a=60, we can see both resolutions and it is clear.

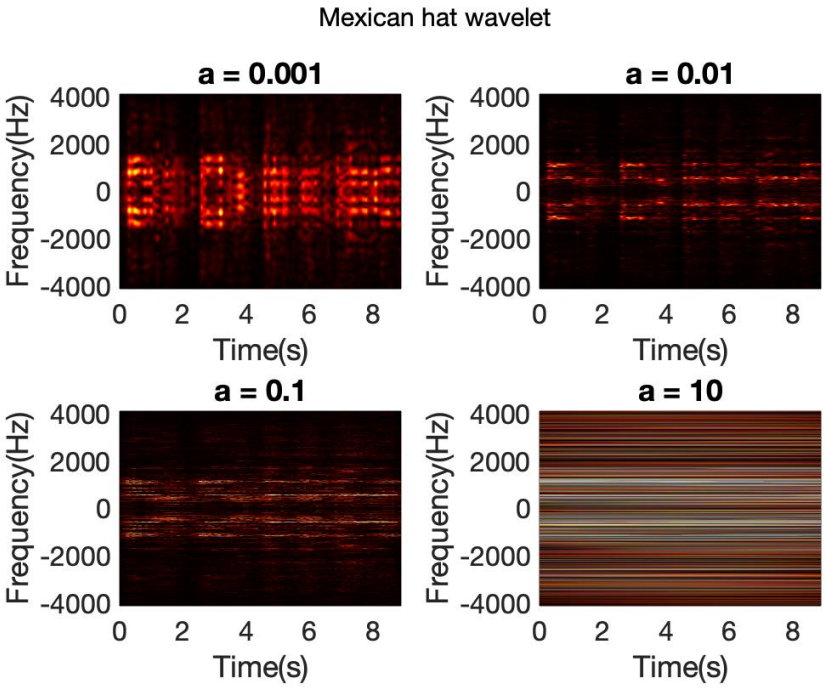

Figure 2.The spectrogram when we apply Mexican har wavelets. We can notice as a increase, the width of the filter increases, so we have more frequency resolution and less time resolution; as a decreases, the width decreases, so we have more time resolution, but less frequency resolution. It seems like a = 0.01 is a good choice since we are able to view both information.

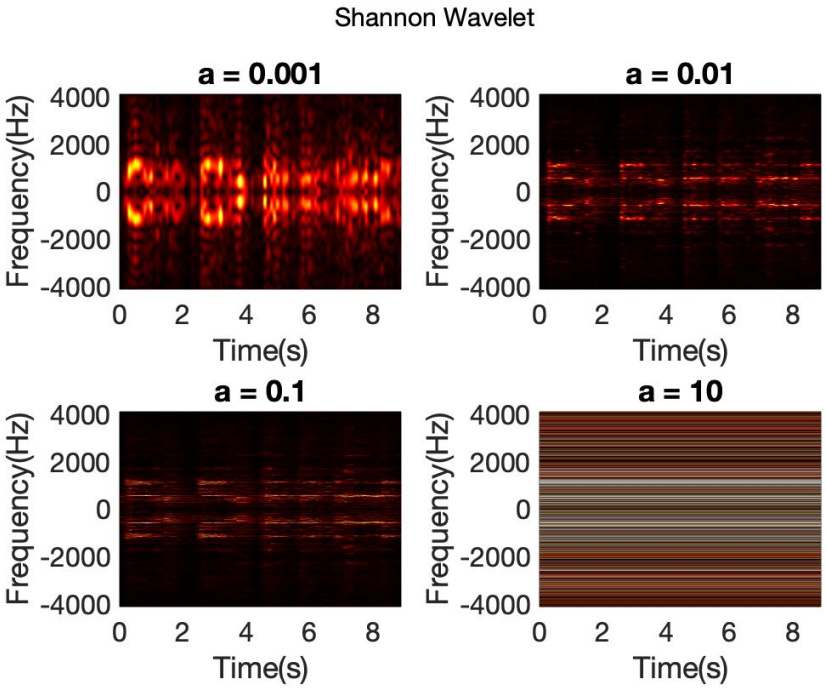

Figure 3. The spectrograms we get after we apply the Shannon wavelet. The graph follows similar pattern as the Mexican hat wavelet.

4.2 Part 2

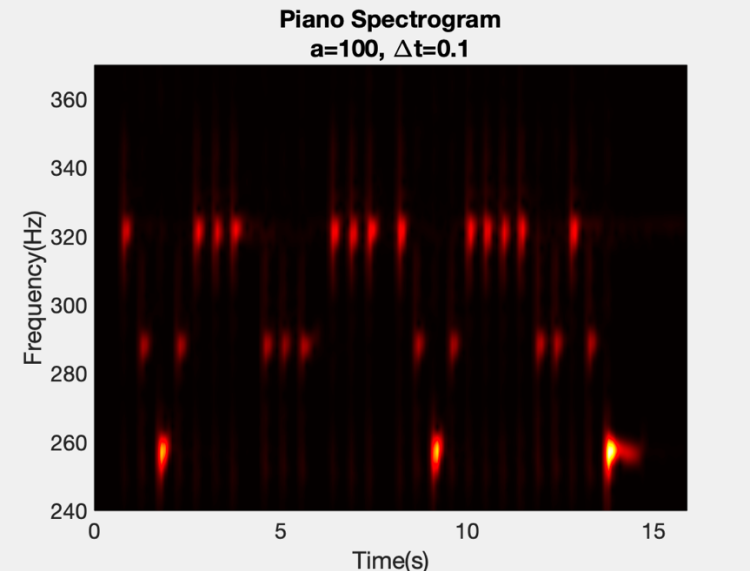

Figure 4. This is the spectrogram of Mary had a little lamb played by piano using gaussian filter. We can see the central frequencies seems smeared, which is because of the overtone above and below them. The overtone appears at the same time as the central frequency. If we correspond the frequencies to music scale figure given in the assignment, we know the notes are :
EDCDEEE DDD EEE EDCDEEE EDDEDC

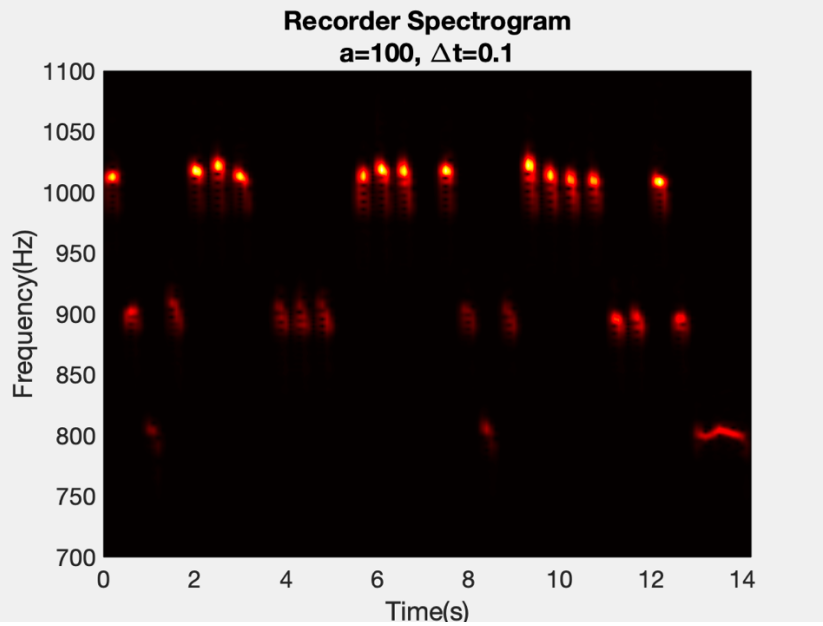

Figure 5. This is the spectrogram of Mary had a little lamb in the recorder after we apply the Gaussian filter. We can see the central frequencies are higher. The overtones are located mainly below the central frequency. This is why we hear these two pieces and know they are played by different instruments.

Sec. V. Summary and Conclusions

To sum up, if given a wavelength, we need to first think about converting angular frequency to radial frequency. Then we define sample frequency, number of points, and total amount of time. Choose a filter function, we multiply it with the FFT of original signal. Then we slide the filter window to the right for Δt, and multiply again. We do the same process until we reach to the end of the time interval. To determine whether the parameters are appropriate, we try several values of transition parameters and scaling parameters to see which value gives a nice spectrogram. Then we can gain both time and frequency information from the spectrogram.

For instruments, Gabor transform is very powerful since the overtone causes problem if we use FFT. So Gabor transform handles overtone since the overtone appears at the same time as the central frequency, we can see what frequencies are central and what are overtone on a appropriate spectrogram.

In part 1, we compare how the change in parameters change the results of spectrogram. By Figure 1, 2, & 3, we know that if choosing appropriate parameters, different filters will give the same results in spectrogram.

In part 2, we are given two pieces of *Mary had a little lamb* -- one played on a piano, one is recorder. By Figure 5 and 6, we can see that their timbre are different because their overtones have different patterns. Additionally, their frequencies are different. Even though the same piece, recorder has higher frequency – bounded by 750 – 1100Hz; whereas the piano has lower frequency – bounded by 240 -370Hz.

Appendix A. MATLAB functions used and brief implementation explanation

1. `pcolor(X,Y,C)` specifies the x- and y-coordinates for the vertices. The size of `C` must match the size of the x-y coordinate grid. For example, if `X` and `Y` define an m-by-n grid, then `C` must be an m-by-n matrix.

2. `colormap(map)` sets the colormap for the current figure to the colormap specified by `map`.

3. `shading interp` varies the color in each line segment and face by interpolating the colormap index or true color value across the line or face.

4. `[s,w,t] = spectrogram(x,window,noverlap,w)` returns the spectrogram at the normalized frequencies specified in `w`.

5. `[y,Fs] = audioread(filename)` reads data from the file named `filename`, and returns sampled data, `y`, and a sample rate for that data, `Fs`.

Appendix B.MATLAB codes

```
load handel
v = y';
plot((1:length(v))/Fs,v);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');

n=73113; L=n./Fs;
t2=linspace(0,L,n+1); t=t2(1:n);
k=(2*pi/L)*[0:(n-1)/2 -(n-1)/2:-1];
```

```
ks=fftshift(k);
vt = fft(v);


%% plot spectrogram of gabor transform of v
figure(1)
a_vec = [0.1 60 100 10000];
tslide=0:0.1:L;
for jj = 1:length(a_vec)
    a = a_vec(jj);
    Sgt_spec = zeros(length(tslide),n);
    Sgt_spec = [];
    for j=1:length(tslide)
        g=exp(-a*(t-tslide(j)).^2);
        Sg=g.*v;
        Sgt=fft(Sg);
        Sgt_spec(j,:) = fftshift(abs(Sgt));
    end

    subplot(2,2,jj)
    pcolor(tslide,ks,Sgt_spec.'),
    shading interp
    title(['a = ',num2str(a)],'Fontsize',16)
    xlabel('Time(s)'),ylabel('Frequency(Hz)')
    set(gca,'Fontsize',16)
    colormap(hot)
end
sgtitle('Gaussian filter','Linewidth',18)

%% Mexican hat wavelet

figure(2)
a_vec = [0.001 0.01 0.1 10];
for jj = 1:length(a_vec)
    a = a_vec(jj);
    tslide=0:0.1:L;
    Sgt_spec = zeros(length(tslide),n);
    Sgt_spec = [];
    for j=1:length(tslide)
        m = 1/sqrt(a).*(1- ( (t-tslide(j)) ./a ).^2).*exp(-( (t-
tslide(j)) ./a).^2./2);
        Sg=m.*v;
        Sgt=fft(Sg);
        Sgt_spec(j,:) = fftshift(abs(Sgt));
    end

    subplot(2,2,jj)
    pcolor(tslide,ks./(2*pi),Sgt_spec.'),
    shading interp
    title(['a = ',num2str(a)],'Fontsize',16)
    xlabel('Time(s)'),ylabel('Frequency(Hz)')
    set(gca,'Fontsize',16)
    colormap(hot)
end
sgtitle('Mexican hat wavelet','Linewidth',16)

%% step function
```

```
figure(3)
a_vec = [0.001 0.01 0.1 10];
for jj = 1:length(a_vec)
    a = a_vec(jj);
    tslide=0:0.1:L;
    Sgt_spec = zeros(length(tslide),n);
    Sgt_spec = [];
    for j=1:length(tslide)
        s = abs(t-tslide(j))<=a;
        Sg=s.*v;
        Sgt=fft(Sg);
        Sgt_spec(j,:) = fftshift(abs(Sgt));
    end

    subplot(2,2,jj)
    pcolor(tslide,ks./(2*pi),Sgt_spec.'),
    shading interp
    title(['a = ',num2str(a)],'Fontsize',16)
    xlabel('Time(s)'),ylabel('Frequency(Hz)')
    set(gca,'Fontsize',16)
    colormap(hot)
end
sgtitle('Shannon Wavelet','Linewidth',16)
```

```
clear; close all; clc

[y1,Fs] = audioread('music1.wav');
v1 = y1';
tr_piano=length(y1)/Fs; % record time in seconds

subplot(2,1,1)
plot((1:length(y1))/Fs,y1);
hold on
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (piano)');

L=tr_piano; n=701440;
t2=linspace(0,L,n+1); t=t2(1:n);
k=(2*pi/L)*[0:n/2-1 -n/2:-1];
ks=fftshift(k);
vt = fft(v1);

subplot(2,1,2)
plot(ks,fftshift(abs(vt)))
xlabel('FFT(Sg)'),ylabel('frequency')

%%
% a decrease, width increase
% delta t big == less time resolution; delta t small, more time resolution
```

```
figure(2)
a = 100;
tslide=0:0.1:L;
Sgt_spec = zeros(length(tslide),n);
Sgt_spec = [];
for j=1:length(tslide)
    g=exp(-a*(t-tslide(j)).^2);
    Sg=g.*v1;
    Sgt=fft(Sg);
    Sgt_spec(j,:) = fftshift(abs(Sgt)); % We don't want to scale it
end

pcolor(tslide,ks./(2*pi),Sgt_spec.')
title({'Piano Spectrogram','a=100, \Deltat=0.1'})
xlabel('Time(s)'),ylabel('Frequency(Hz)')
shading interp
set(gca,'Ylim',[230 370],'Fontsize',16)
colormap(hot)
```

```
clear; close all; clc

[y1,Fs] = audioread('music2.wav');
v1 = y1';
tr_piano=length(y1)/Fs; % record time in seconds

subplot(2,1,1)
plot((1:length(y1))/Fs,y1);
hold on
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (recorder)');

L=tr_piano; n=length(y1);
t2=linspace(0,L,n+1); t=t2(1:n);
k=(2*pi/L)*[0:n/2-1 -n/2:-1];
ks=fftshift(k);
vt = fft(v1);

subplot(2,1,2)
plot(ks,fftshift(abs(vt)))
xlabel('FFT(Sg)'),ylabel('frequency')

%%
% a decrease, width increase
% delta t big == less time resolution; delta t small, more time resolution

figure(2)
a = 100;
tslide=0:0.1:L;
Sgt_spec = zeros(length(tslide),n);
Sgt_spec = [];
for j=1:length(tslide)
```

```
    g=exp(-a*(t-tslide(j)).^2);
    Sg=g.*v1;
    Sgt=fft(Sg);
    Sgt_spec(j,:) = fftshift(abs(Sgt)); % We don't want to scale it
end

pcolor(tslide,ks./(2*pi),Sgt_spec.')
title({'Recorder Spectrogram','a=100, \Deltat=0.1'})
xlabel('Time(s)'),ylabel('Frequency(Hz)')
shading interp
set(gca,'Ylim',[700 1100],'Fontsize',16)
colormap(hot)
```