AMATH 582 Homework 2: Gábor Transforms

Sara L. Daley

2/7/2020

Abstract

The goal of this paper is to analyze music using Gábor Transforms or Short-Time Fourier Transforms. First we analyze a small portion of Handel's Messiah and look at its spectrogram using a few different filters. Then we look at a composure of Mary Had a Little Lamb on both a piano and recorder and compare the two pieces.

1 Introduction and Overview

This code set has two parts. The first part is to take a sample of Handel's Messiah and filter it using Short-Time Fourier Transforms with filters: Gaussian, Step, and Mexican-Hat. For Gaussian we dive deeper to various filter widths and time steps. We can compare the effectiveness of each filter on this piece by looking at each spectrogram. A spectrogram is simply a way to represent a signal in time and frequency. The second part of this homework is to analyze the song 'Mary Had a Little Lamb' played on both the piano and recorder. We use a Gaussian filter to recreate the music score for both pieces and take a look at each notes' resulting frequencies.

2 Theoretical Background

The Fourier Transform (FT) is vital when it comes to time-frequency analysis. However, the drawback of the FT is that we can solve for the frequency

of the signal, but lose time resolution[1]. In essence, it is the opposite drawback of time series which loses frequency resolution of the signal in the time domain. The Gábor transform,

$$\mathcal{G}[f](t,w) = \tilde{f}_g(t,w) = \int_{-\infty}^{\infty} f(\tau)g(\tau - t)e^{-i\omega\tau}d\tau \tag{1}$$

where $g(\tau - t)$ induces localization of the Fourier integral around $t = \tau$[1]. Equation 1 localizes both time *and* frequency by creating the *short-time Fourier transform (STFT)*[1]. See Figure 1 for a nifty visualization on each of the three methods. In order to achieve the Gábor transform, we discretize

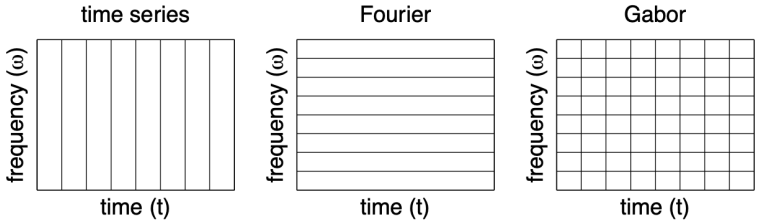


Figure 1: Difference between time series, Fourier, and Gábor[1]

in the time and frequency domains. There are of course are trade offs with this method as well in that one can sample using a filtering window that is wide, which you gain more frequency resolution while diminishing time resolution. One could go the other way as well and use a filter so narrow that it captures time resolution well but not frequency. Thus, it is a balancing act that will depend on the signal itself. In order to do this we choose a filter and apply it to a signal while sliding in time. In this code we implement Gábor Transform methods using these filters: Gaussian, Step (or referred to as Shannon), and Mexican-Hat.

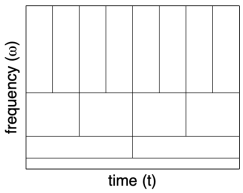


Figure 2: time-frequency resolution achieved with the wavelet transform[1]

The key here is that Gábor handles translation (τ) and scaling (a) but are fixed and to fix them may mean cutting out key information when we have a varying signal in time. This leads to using wavelets, where we can decrease scale a in order to pull out smaller frequency waves in a signal. Figure 2 adds to Figure 1 to show how a wavelet transform can produce both time and frequency resolution. This means that wavelets can be more than just a filter; they can be used as a formal signal decomposition.

3 Algorithm Implementation and Development

Please refer to Algorithm 1

Algorithm 1: Short-Time Fourier Transform Algorithm

Import Signal from `Handel` or `music.wav`
Set *Frequency sample rate* (`Fs`), *Fourier Modes* (`n`)
Define domain discretization (shifted) (`k`)
`fftshift` domain to provide unshifted counterpart (`ks`)
Set time slide `0:step:signal-time-length`
for $j = 1 : length(TimeSlide)$ **do**
  Define Filter, regarding Fourier kernel $g(\tau - t)$
  $Filter \times Signal$
  `fft` $Filter \times Signal$ result
  `abs(fftshift(fft(`$Signal \times Filter$`) result))` {*This step solves the Fourier Transform at each time slice*}
  create Matrix using previous result
end for
plot spectrogram using `pcolor`

4 Computational Results

The first result we consider is within Handel's Messiah, using the Gábor Transform (the Gaussian filter). Recall that a Gaussian is,

$$G = e^{-\tau(x-x_0)^2+(y-y_0)^2} \tag{2}$$

where τ is the filter width. Therefore, the larger τ, the more narrow the filter width. We tried various filter widths to see how that changed the spectrogram. In Figure 3 we see what the different widths look like and can gather how much signal passes through them, and Figure 4 shows the resulting spectrograms with those widths. It is clear from the spectrograms that with width 10, we have good frequency resolution but hardly any time resolution. In widths 200 and 400 we begin to see both frequency and time resolution. We chose to stick with filter width 400.

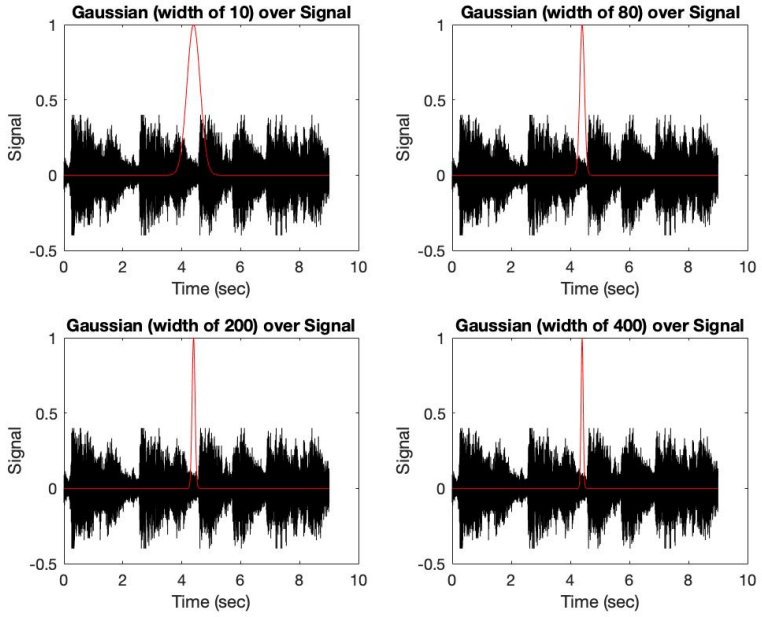


Figure 3: Various widths of the Gaussian filter over Handel Signal

The next result to consider is varying the time steps. We used time steps of 1.0, 0.5, 0.1, and 0.01 all with filter width 400. Figure 5 shows the resulting spectrogram of changing the time steps. We see that with a time step of 1.0, we have good time resolution but bad frequency resolution. This implies that we are undersampling the data. The other end of the test used steps of 0.01. Though this shows we have a clear time and frequency resolution-leading to oversampling. Oversampling isn't the worst thing to do but does have computational drawbacks...running this code took much longer and it was decided to use step 0.1 for the rest of programming.

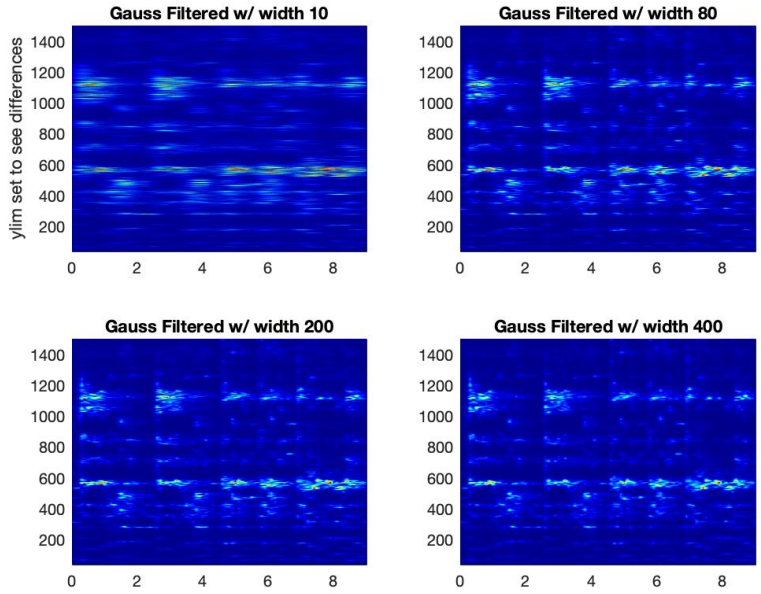


Figure 4: Signal spectrograms using varying widths of the Gaussian filter

Next we consider different Gábor windows. For all windows we used a time step of 0.1. We created spectrograms for the Gaussian filter (Figure 6), Step (or Shannon) filter (Figure 7), and the Mexican-Hat (Figure 8). All filters had frequency and time resolution, though it was easier to get the Gabor and Step function to have the best resolution. It is clear looking at each filter that with a different signal, entirely different conclusions about the filters could be made.

The second part of the code was to analyze the song, 'Mary Had a Little Lamb' on both the piano and recorder. To filter I used a Gaussian with width 30 for the piano and 20 for the recorder to isolate the music score. Figure 9 shows the spectrogram for the two different instruments. A horizontal line was drawn at each of the notes with the frequency (in Hz) the line was drawn at. For the piano, it is to be notes E, D, C with C being middle C. The actual frequencies of those notes are 329.63, 293.66, 261.63 Hz and we see them at about 320, 285, 255. This may indicate that the piano is out of tune. With the piano we notice more noise at each of the note frequencies and this is due to overtones. Figure 9 did not filter out the overtones as the notes were clear

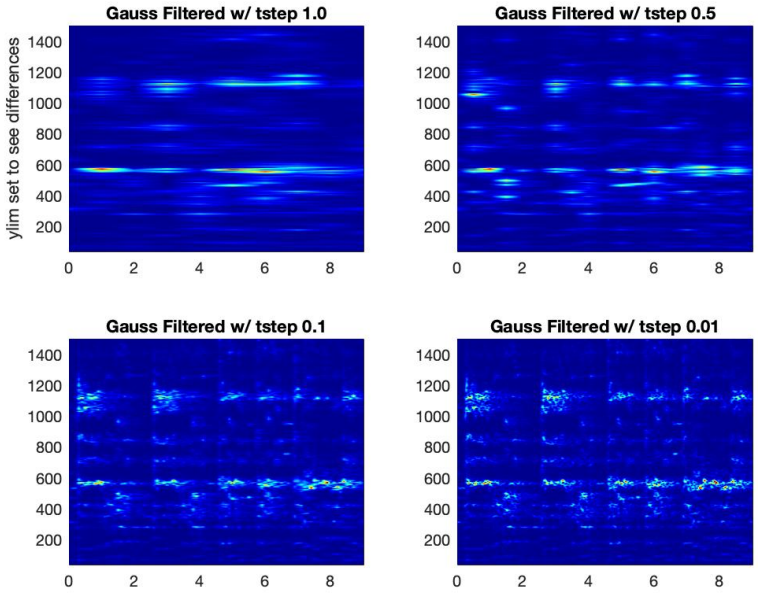


Figure 5: Signal spectrograms using varying time steps

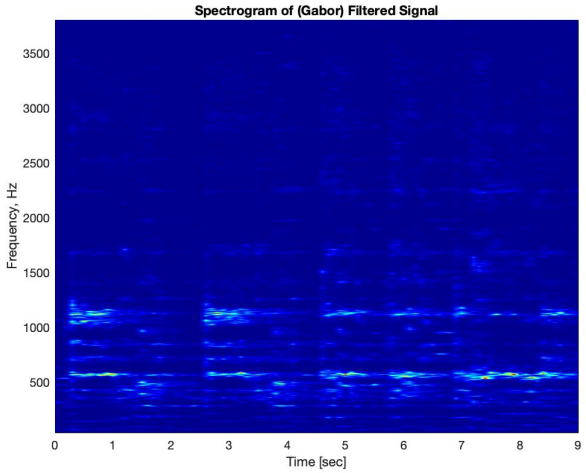


Figure 6: Signal spectrogram using Gaussian Filter

in the spectrogram. To filter the overtones you could take the average of the

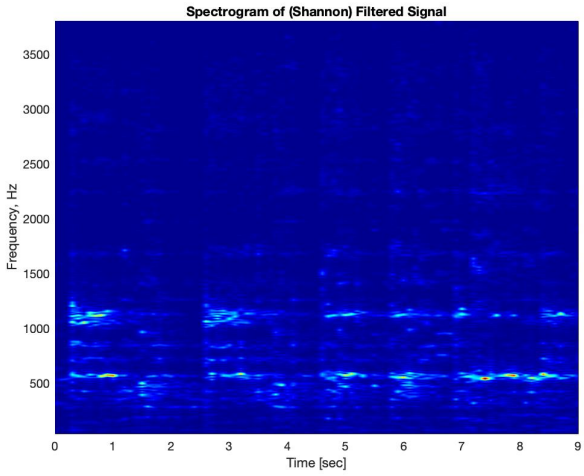


Figure 7: Signal spectrogram using Step Filter

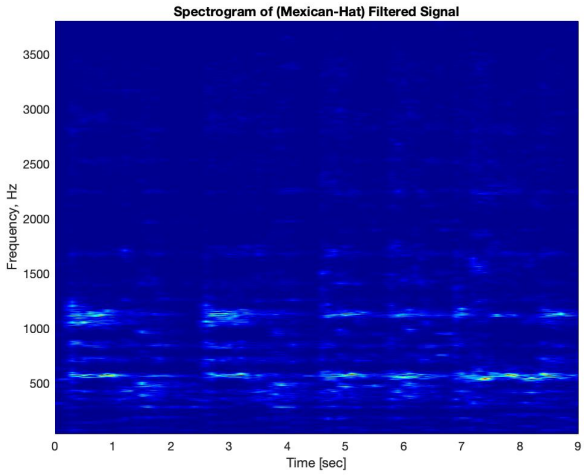


Figure 8: Signal spectrogram using Mexican-Hat Filter

signal for that time window. For the recorder, it is to notes B, A, G. The actual frequencies for those notes are 987.77, 880.00, 783.99 Hz and we see them at about 905, 800, 715. Though the difference of frequencies of these notes is greater than with the piano, it could be simply the way the recorder is being played, i.e., the tone would rely on the pressure going into it.

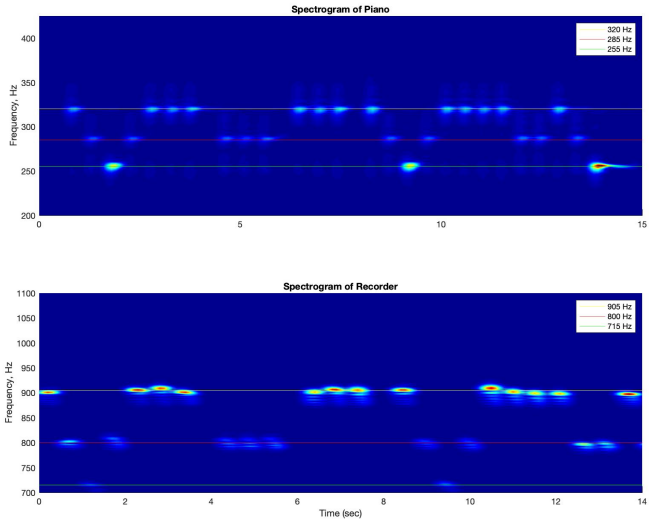


Figure 9: Signal spectrogram of the piano and recorder, isolating music notes

5 Summary and Conclusions

To analyze Handel's 'Messiah' we first looked at various Gaussian filter widths. The goal was to choose a filter width that captured good signal resolution. While the smaller the filter width did take more steps to move across the music, it offered the best resolution of the music. We then looked at the effect of the step, and started with a large step of one second. In Figure 5 we saw that the smallest time step of 0.01 offered great resolution due to oversampling, it took too long to run the code and a step of 0.1 was chosen. The last analysis we did with Handel's 'Messiah' was trying three different filters: Gaussian, Shannon (or step), and Mexican-Hat. All filters showed good time and frequency resolution; however, the Gaussian and Shannon filters showed the best resolutions. The last part of the homework was to analyze 'Mary Had a Little Lamb' on both the recorder and piano. We were able with a Gaussian filter to isolate each note played and the resulting frequency.

References

[1] Jose Nathan Kutz. *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*, Oxford University Press, 2013.

[2] `https://www.mathworks.com/help/matlab/index.html`

Appendices

Please refer to my Github repo for all code and related documents.

A MATLAB Functions

If there is a MATLAB Function explained in a previous homework set, I do not go into detail here. Therefore, if a function is seen in the MATLAB code that is not explained here- refer to previous homework sets.

- Transpose a vector: $v = y'$, makes a column vector a row and vice versa.
- `[Y, Fs]=audioread(FILENAME)`

  "Reads an audio file specified by the character vector or string scalar `FILENAME`, returning the sampled data in Y and the sample rate FS, in Hertz." [2]
- `audioplayer(Y, Fs)`

  "creates an audioplayer object for signal Y, using sample rate Fs." [2] Note, a handle to the object is returned
- `playblocking(OBJ)` plays back the `audioplayer` object (created in the previous listed function.
- `plot(x,y)`

  `plot` is self explanatory. However here are some commonly used functions to go with plot:

  - `xlabel('String of x-axis label')` repeat for `ylabel`

- `title('String of Plot title')`
- `subplot(row,col,NumPlot)`
- `pcolor(x,y,c)` size of x,y,c are important:

  Ex: `size(x)` = $1by91$, `size(y)`= $1by73112$, `size(c)`= $91by73112$. **Dimensions must match!**[2]

- `colormap`

  colormap sets the color of the spectrogram. Used `jet` in this code, default is `parula`, these are also good options: `hot`, `cool`, `gray` (grayscales it- perfect for printing). If colors `hot` and `cool` are what you're looking for but aren't quite right- consider `autumn` and `winter`.

B MATLAB Codes

Table of Contents

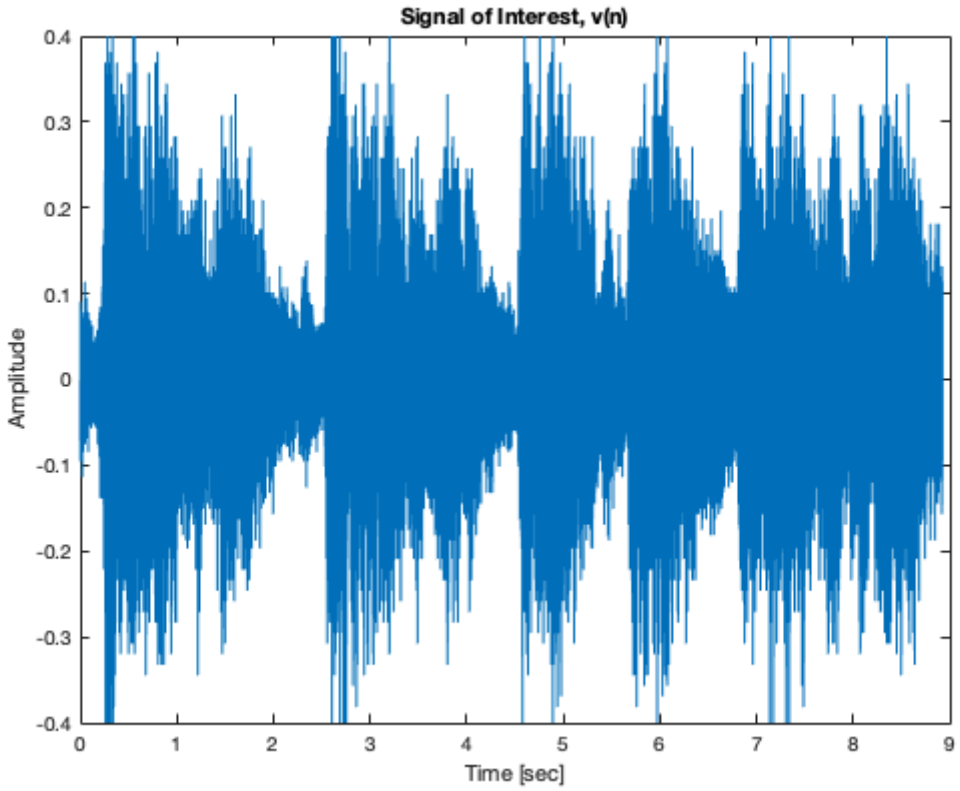

```
clear all; close all; clc;
```

Load and Initialize for Handel

```
load handel
v = y'/2;
v = v(1:end-1); %v_end=0 so cut it out to make v even in length

L=9; % spatial domain
n=length(v); % Fourier modes
t=linspace(0,L,n);
k=(2*pi/L)*[0:n/2-1 -n/2:-1]; % Freq space rescaled to 2pi domain
ks=fftshift(k); % Unshift k

S = v;
St = fft(S);
tslide = 0:0.1:9;
```

Handel Signal

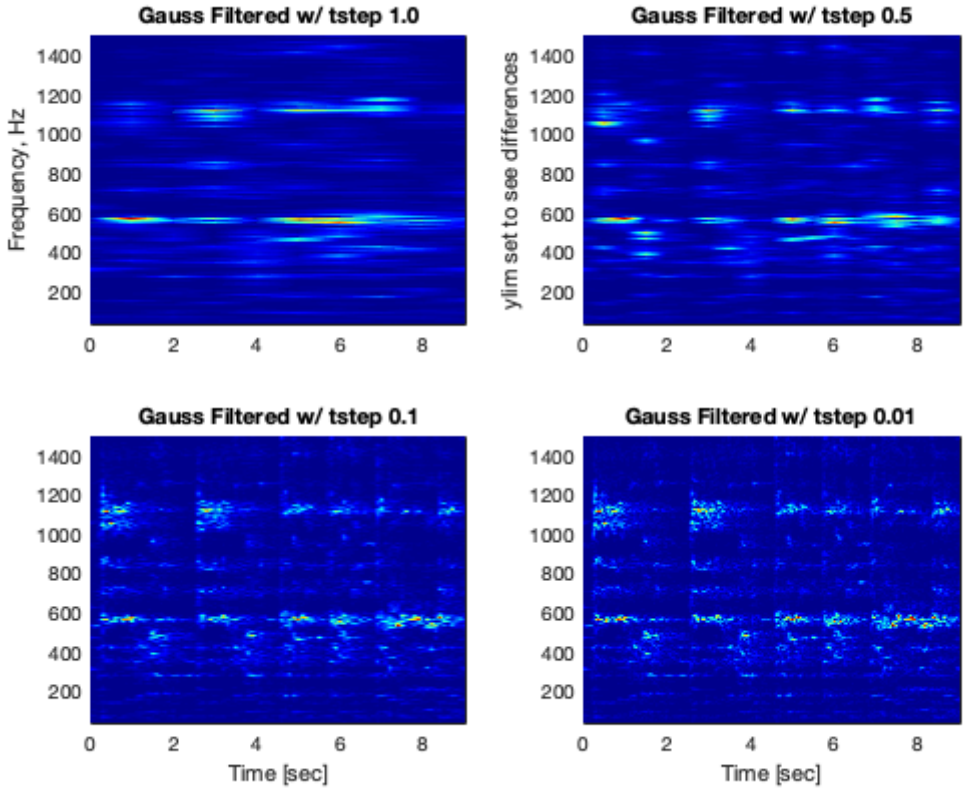
```
plot((1:length(v))/Fs,v);
xlabel('Time [sec]')
ylabel('Amplitude')
title('Signal of Interest, v(n)')

%p8 = audioplayer(v,Fs);
%playblocking(p8);
```



Try Various Time Slide Step Sizes

```
slides = {0:1:9 0:0.5:9 0:0.1:9 0:0.01:9};
for k=1:length(slides)
    Sgt_spec{k} = [];
    tslide = slides{k};
    for j=1:length(tslide)
        g = exp(-400*(t-tslide(j)).^2); % Gabor
        Sg = g .* S;
        Sgt = fft(Sg);
        Sgt_spec{k} = [Sgt_spec{k}; abs(fftshift(Sgt))];
    end
end

figure(15)
subplot(2,2,1)
pcolor(slides{1}, ks/(2*pi), Sgt_spec{1}.'), shading interp
colormap(jet)
ylim([38 1500])
ylabel('Frequency, Hz')
title('Gauss Filtered w/ tstep 1.0')
subplot(2,2,2)
pcolor(slides{2}, ks/(2*pi), Sgt_spec{2}.'), shading interp
colormap(jet)
ylim([38 1500])
```

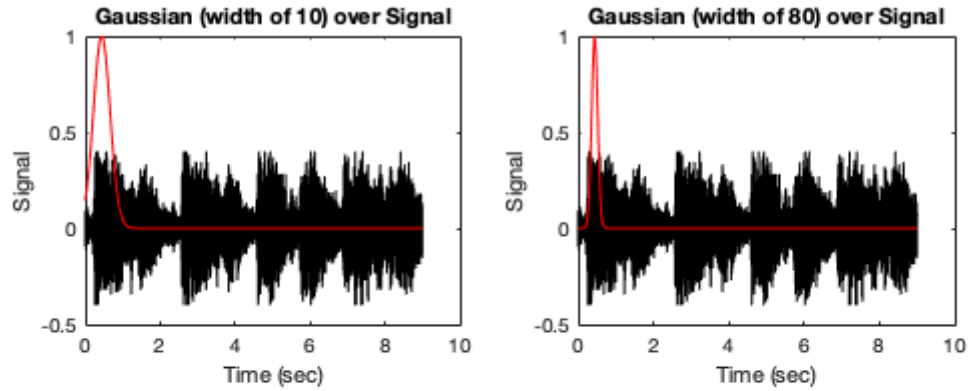
```
ylabel('ylim set to see differences')
title('Gauss Filtered w/ tstep 0.5')
subplot(2,2,3)
pcolor(slides{3}, ks/(2*pi), Sgt_spec{3}.'), shading interp
colormap(jet)
ylim([38 1500])
xlabel('Time [sec]')
title('Gauss Filtered w/ tstep 0.1')
subplot(2,2,4)
pcolor(slides{4}, ks/(2*pi), Sgt_spec{4}.'), shading interp
colormap(jet)
ylim([38 1500])
xlabel('Time [sec]')
title('Gauss Filtered w/ tstep 0.01')
```



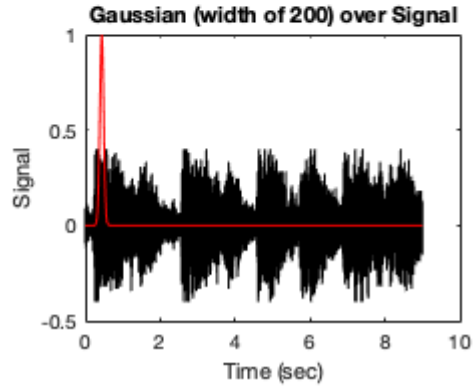
Try Various Gaussian Widths

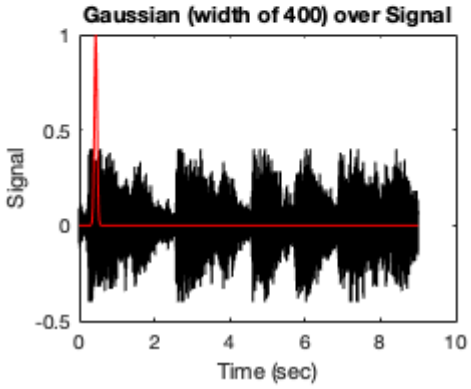
```
a = [10 80 200 400];
for k=1:length(a)
    Sgt_spec{k} = [];
    for j=1:length(tslide)
        g = exp(-a(k)*(t-tslide(j)).^2); % Gabor
        Sg = g .* S;
        Sgt = fft(Sg);
        Sgt_spec{k} = [Sgt_spec{k}; abs(fftshift(Sgt))];
    end
```

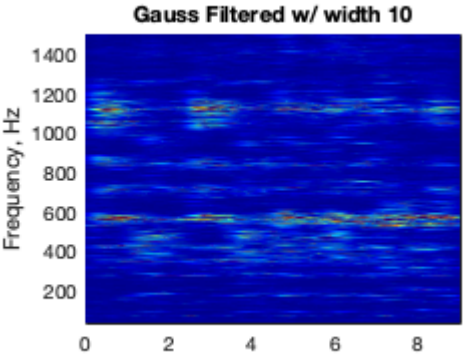
```
end

figure(13)
subplot(2,2,1)
plot(t, S, 'k', t, exp(-10*(t-tslide(45)).^2), 'r')
title('Gaussian (width of 10) over Signal')
xlabel('Time (sec)')
ylabel('Signal')
subplot(2,2,2)
plot(t, S, 'k', t, exp(-80*(t-tslide(45)).^2), 'r')
title('Gaussian (width of 80) over Signal')
xlabel('Time (sec)')
ylabel('Signal')
subplot(2,2,3)
plot(t, S, 'k', t, exp(-200*(t-tslide(45)).^2), 'r')
title('Gaussian (width of 200) over Signal')
xlabel('Time (sec)')
ylabel('Signal')
subplot(2,2,4)
plot(t, S, 'k', t, exp(-400*(t-tslide(45)).^2), 'r')
title('Gaussian (width of 400) over Signal')
xlabel('Time (sec)')
ylabel('Signal')

figure(14)
subplot(2,2,1)
pcolor(tslide, ks/(2*pi), Sgt_spec{1}.'), shading interp
colormap(jet)
ylim([38 1500])
ylabel('Frequency, Hz')
title('Gauss Filtered w/ width 10')
subplot(2,2,2)
pcolor(tslide, ks/(2*pi), Sgt_spec{2}.'), shading interp
colormap(jet)
ylim([38 1500])
ylabel('ylim set to see differences')
title('Gauss Filtered w/ width 80')
subplot(2,2,3)
pcolor(tslide, ks/(2*pi), Sgt_spec{3}.'), shading interp
colormap(jet)
ylim([38 1500])
xlabel('Time [sec]')
title('Gauss Filtered w/ width 200')
subplot(2,2,4)
pcolor(tslide, ks/(2*pi), Sgt_spec{4}.'), shading interp
colormap(jet)
ylim([38 1500])
xlabel('Time [sec]')
title('Gauss Filtered w/ width 400')
```

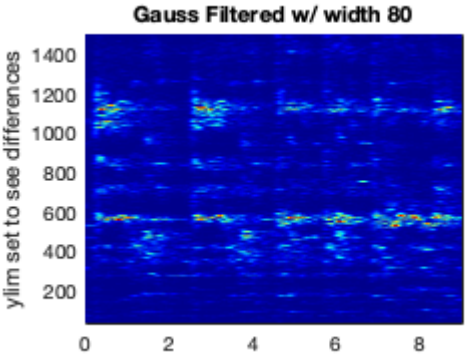

Gaussian (width of 10) over Signal
Signal
Time (sec)

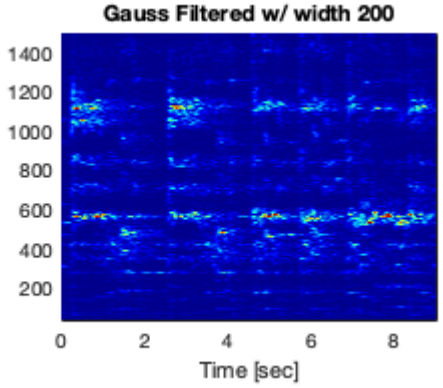
Gaussian (width of 80) over Signal
Signal
Time (sec)

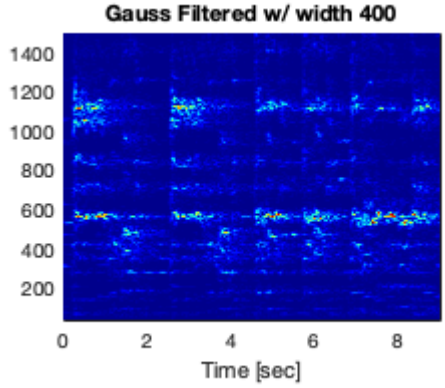

Gaussian (width of 200) over Signal
Signal
Time (sec)


Gaussian (width of 400) over Signal
Signal
Time (sec)


Gauss Filtered w/ width 10
Frequency, Hz

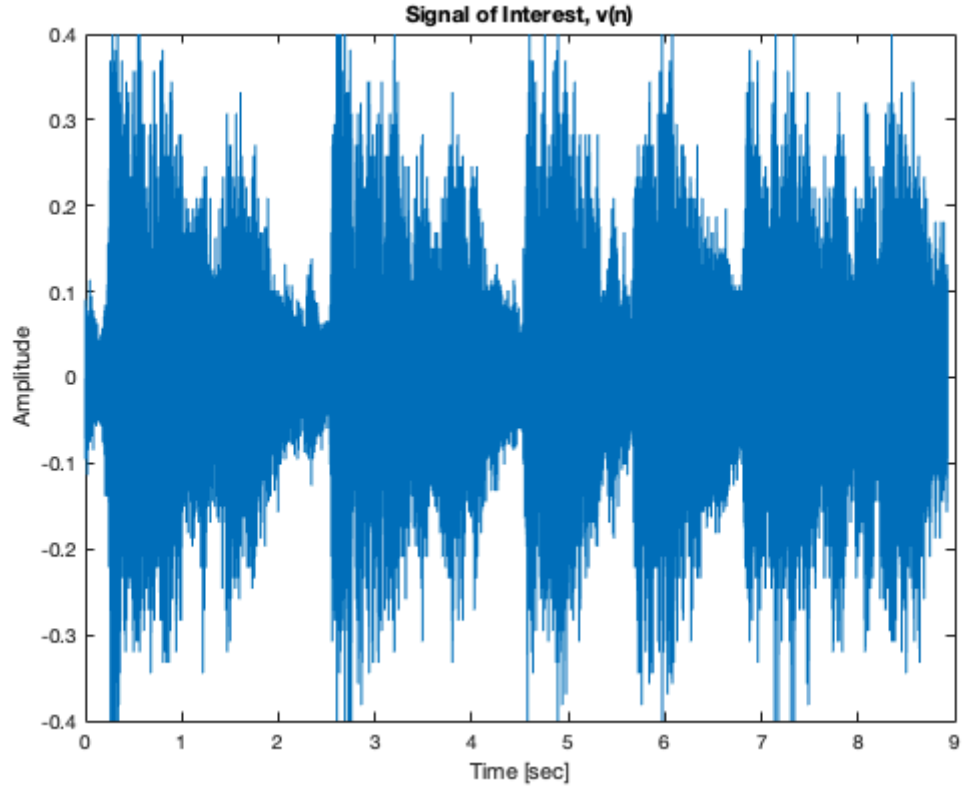

Gauss Filtered w/ width 80
ylim set to see differences

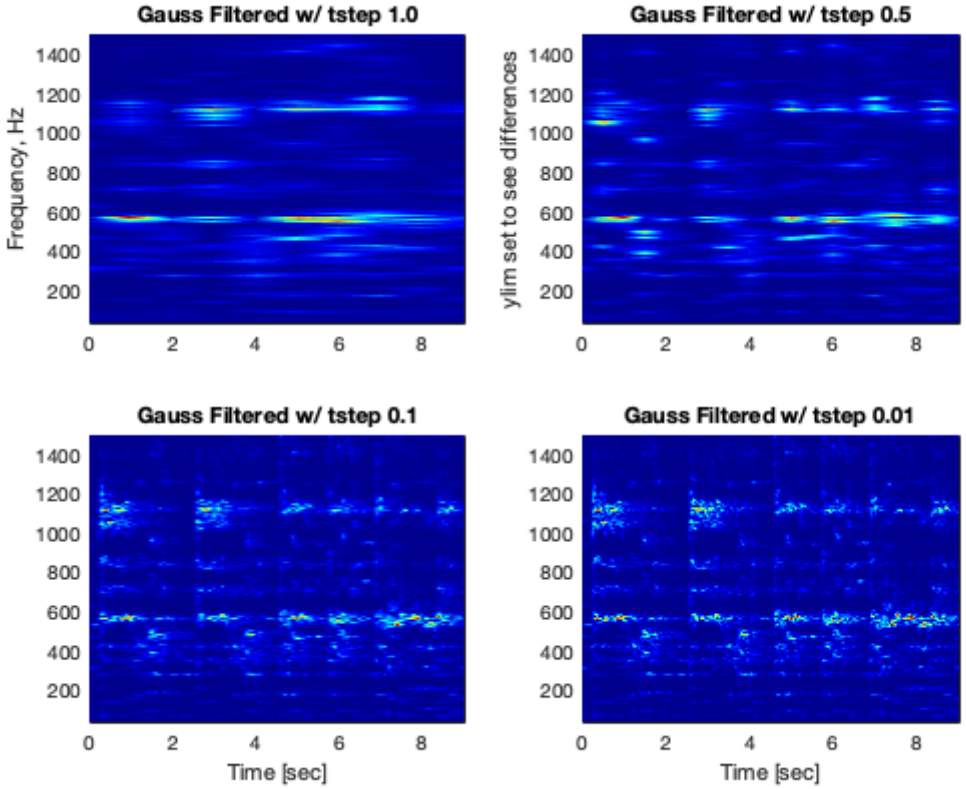

Gauss Filtered w/ width 200
Time [sec]

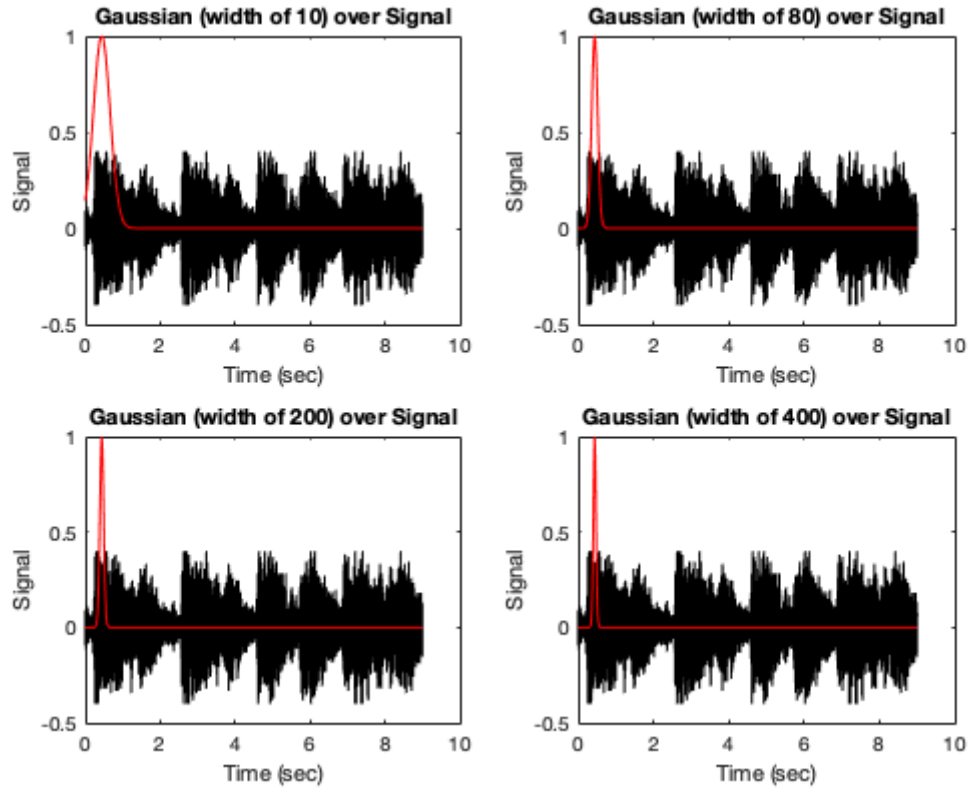

Gauss Filtered w/ width 400
Time [sec]

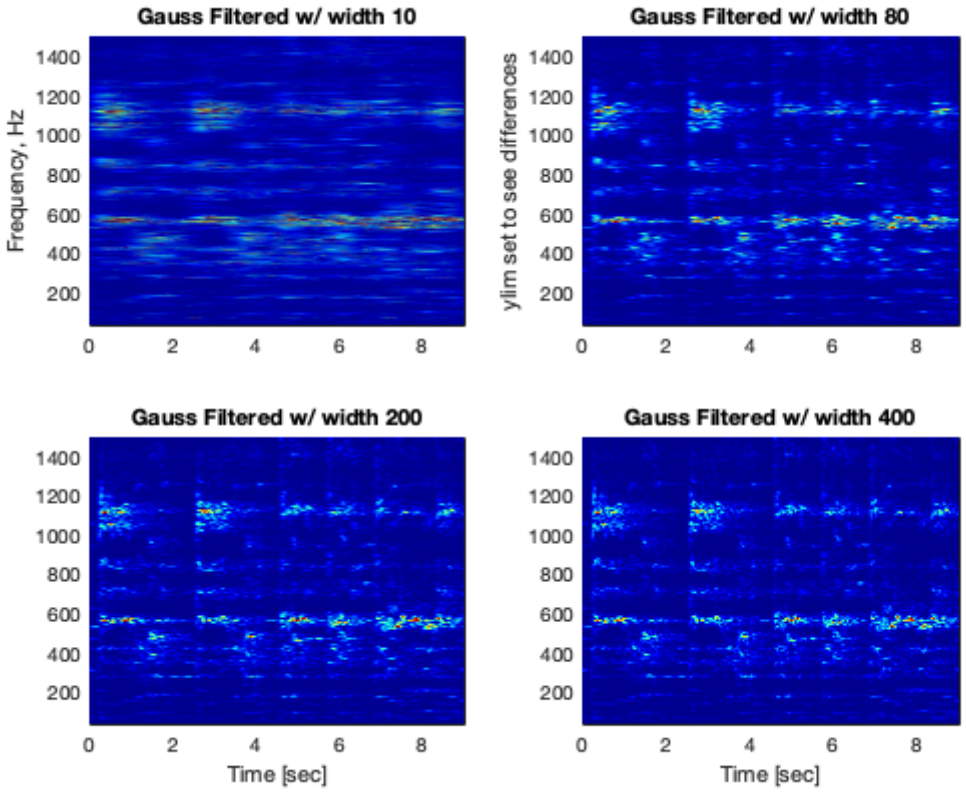
Gabor Transform, Gaussian Filter

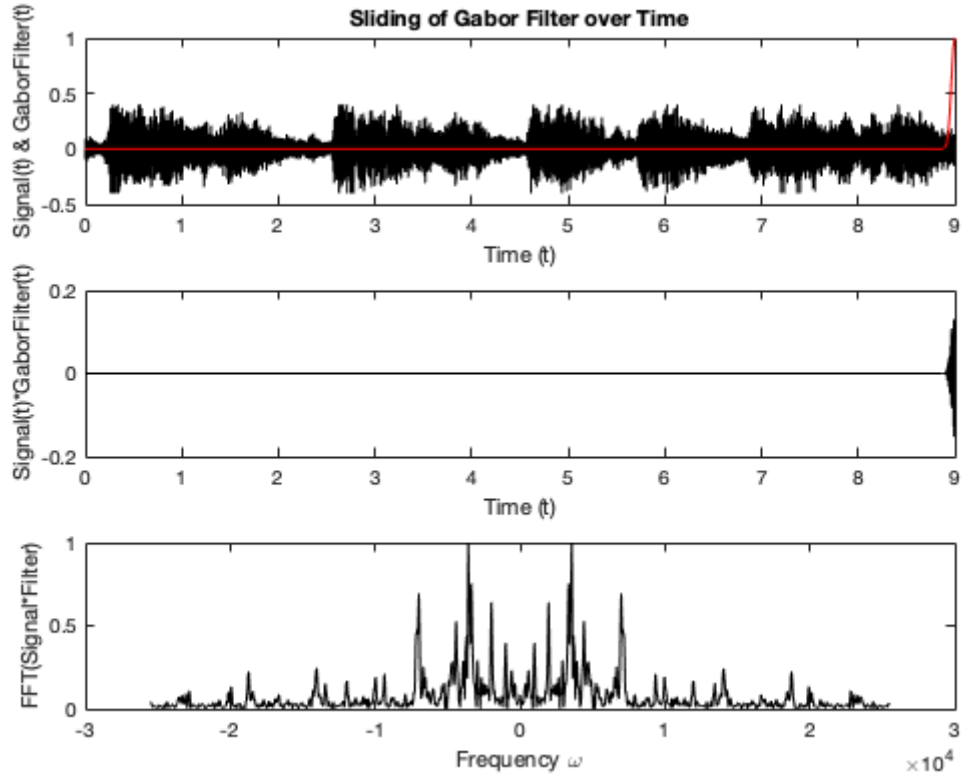
```
Sgt_spec = [];
for j=1:length(tslide)
    g = exp(-400*(t-tslide(j)).^2); % Gaussian
    Sg = g .* S;
    Sgt = fft(Sg);
    Sgt_spec = [Sgt_spec; abs(fftshift(Sgt))];

    figure(2)
    subplot(3,1,1)
    plot(t, S, 'k', t, g, 'r')
    xlabel('Time (t)'), ylabel('Signal(t) & GaborFilter(t)')
    title('Sliding of Gabor Filter over Time')
    subplot(3,1,2)
    plot(t, Sg, 'k')
    xlabel('Time (t)'), ylabel('Signal(t)*GaborFilter(t)')
    subplot(3,1,3)
    plot(ks, abs(fftshift(Sgt))/max(abs(Sgt)), 'k')
    xlabel('Frequency \omega'), ylabel('FFT(Signal*Filter)')
    drawnow
    pause(0.00001)
end
%change y in plot to be Hz.
figure(3)
pcolor(tslide, ks/(2*pi), Sgt_spec.'), shading interp
colormap(jet)
ylim([38 3800])
xlabel('Time [sec]')
ylabel('Frequency, Hz')
title('Spectrogram of (Gabor) Filtered Signal')
```

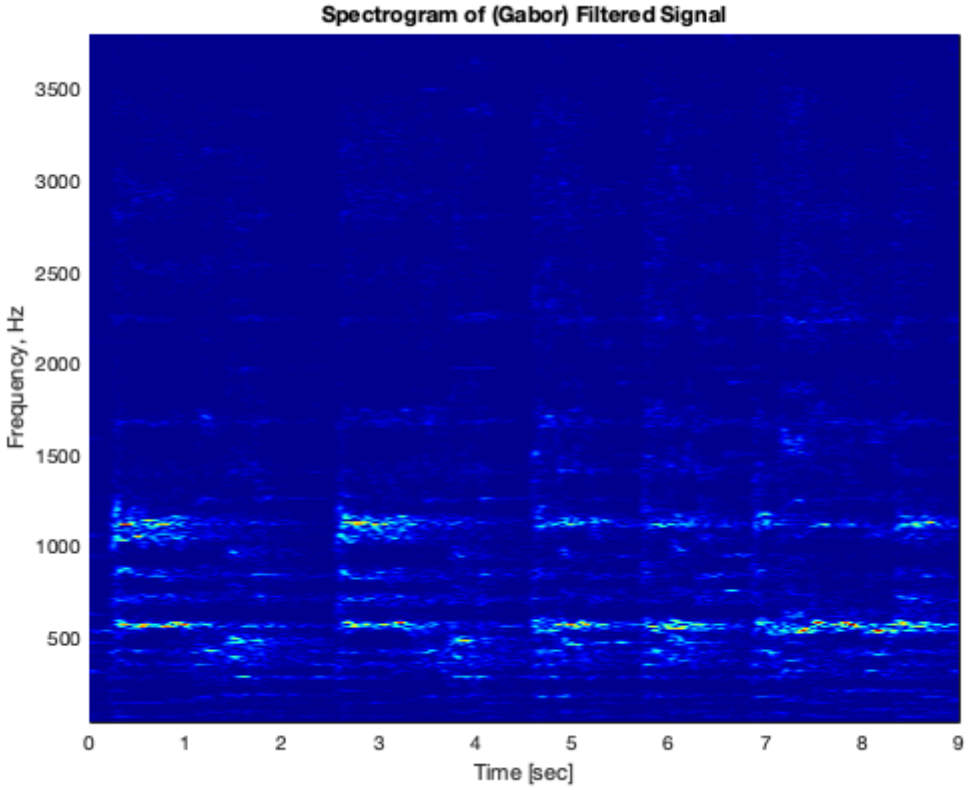

Signal of Interest, v(n)
Amplitude
Time [sec]


Gauss Filtered w/ tstep 1.0
Gauss Filtered w/ tstep 0.5
Gauss Filtered w/ tstep 0.1
Gauss Filtered w/ tstep 0.01
Frequency, Hz
ylim set to see differences
Time [sec]
Time [sec]


Gaussian (width of 10) over Signal
Gaussian (width of 80) over Signal
Gaussian (width of 200) over Signal
Gaussian (width of 400) over Signal
Signal
Time (sec)

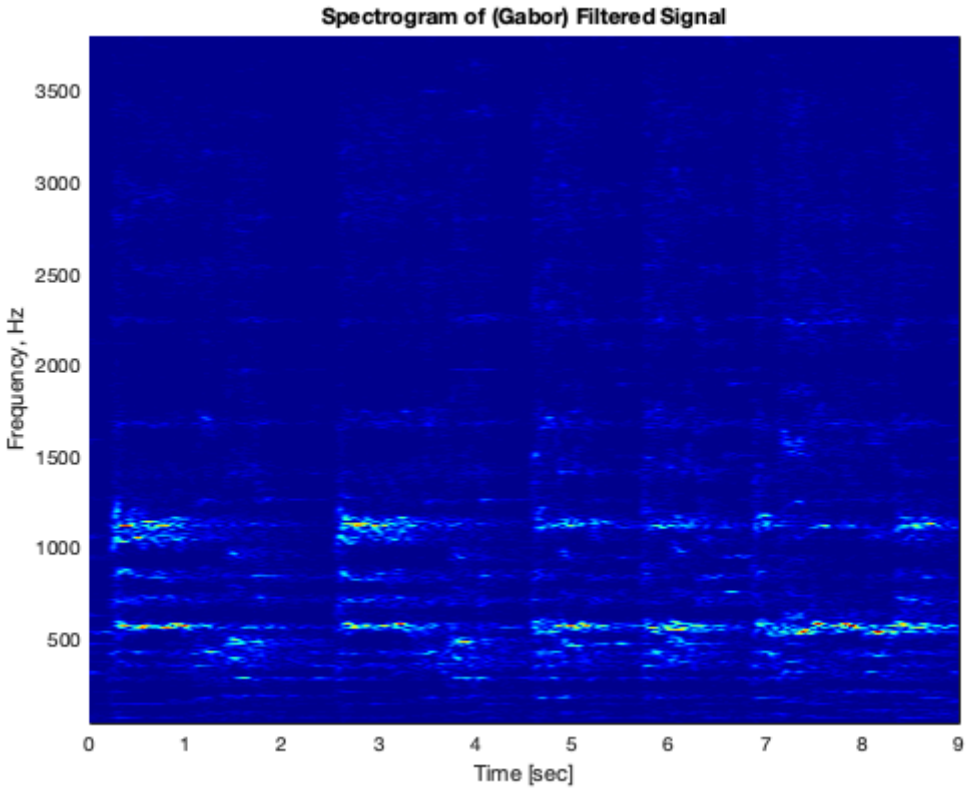

Gauss Filtered w/ width 10
Gauss Filtered w/ width 80
Gauss Filtered w/ width 200
Gauss Filtered w/ width 400
Frequency, Hz
ylim set to see differences
Time [sec]

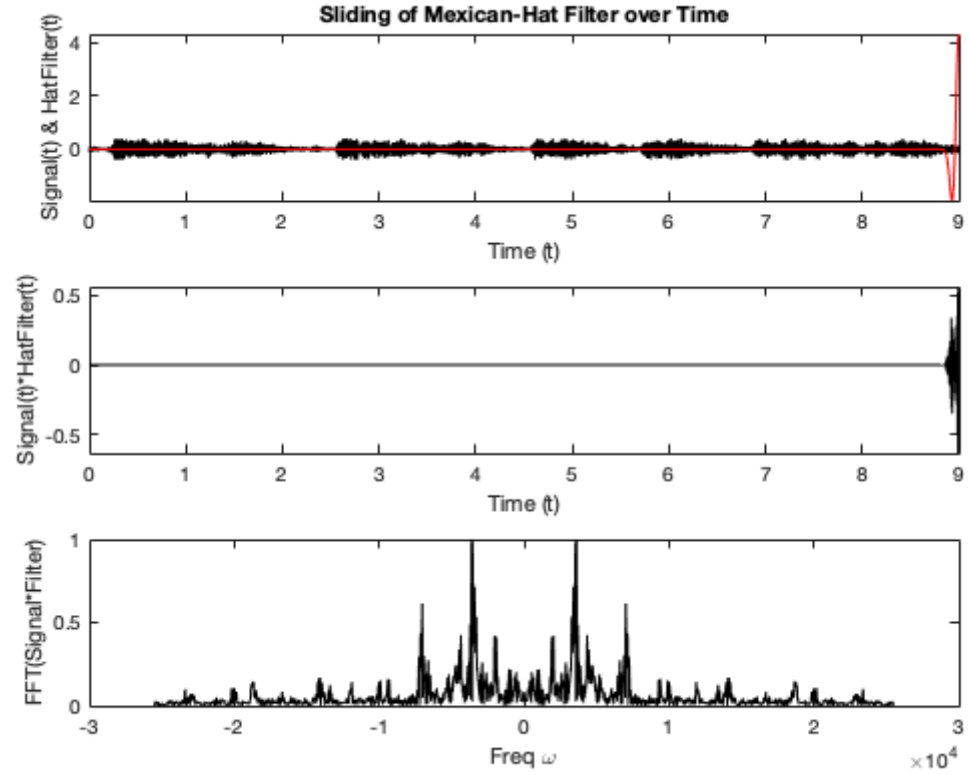

Sliding of Gabor Filter over Time
Signal(t) & GaborFilter(t)
Time (t)
Signal(t)*GaborFilter(t)
Time (t)
FFT(Signal*Filter)
Frequency ω
×10^4

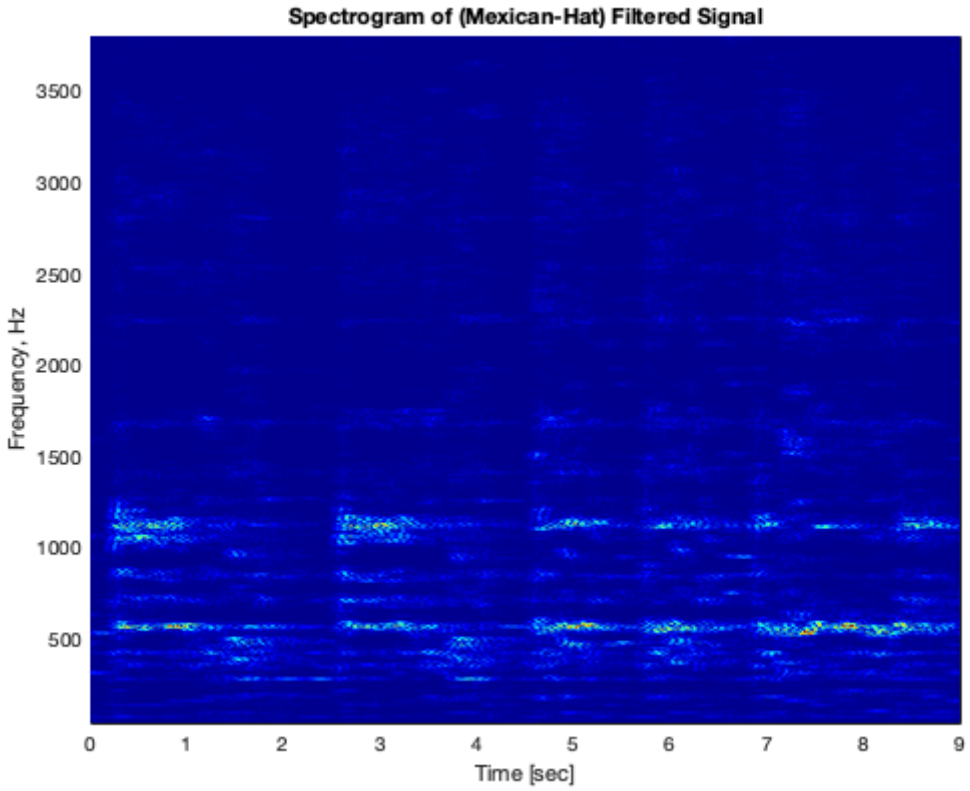

Spectrogram of (Gabor) Filtered Signal
Frequency, Hz
Time [sec]

Gabor Transform, Mexican-Hat Filter

```
Sgt_spec = [];
for j=1:length(tslide)
    g = (2/(sqrt(3*0.04)*pi^(0.25)))* ...
        (1-((t-tslide(j))/0.04).^2).* ...
        exp(-(t-tslide(j)).^2/(2*0.04^2)); % Mexican-Hat
    Sg = g .* S;
    Sgt = fft(Sg);
    Sgt_spec = [Sgt_spec; abs(fftshift(Sgt))];

    figure(4)
    subplot(3,1,1)
    plot(t, S, 'k', t, g, 'r')
    xlabel('Time (t)'), ylabel('Signal(t) & HatFilter(t)')
    title('Sliding of Mexican-Hat Filter over Time')
    subplot(3,1,2)
    plot(t, Sg, 'k')
    xlabel('Time (t)'), ylabel('Signal(t)*HatFilter(t)')
    subplot(3,1,3)
    plot(ks, abs(fftshift(Sgt))/max(abs(Sgt)), 'k')
    xlabel('Freq \omega'), ylabel('FFT(Signal*Filter)')

    drawnow
    pause(0.00001)
end

figure(5)
pcolor(tslide, ks/(2*pi), Sgt_spec.'), shading interp
colormap(jet)
ylim([38 3800])
xlabel('Time [sec]')
ylabel('Frequency, Hz')
title('Spectrogram of (Mexican-Hat) Filtered Signal')
```


Spectrogram of (Gabor) Filtered Signal
Frequency, Hz
3500
3000
2500
2000
1500
1000
500
0 1 2 3 4 5 6 7 8 9
Time [sec]

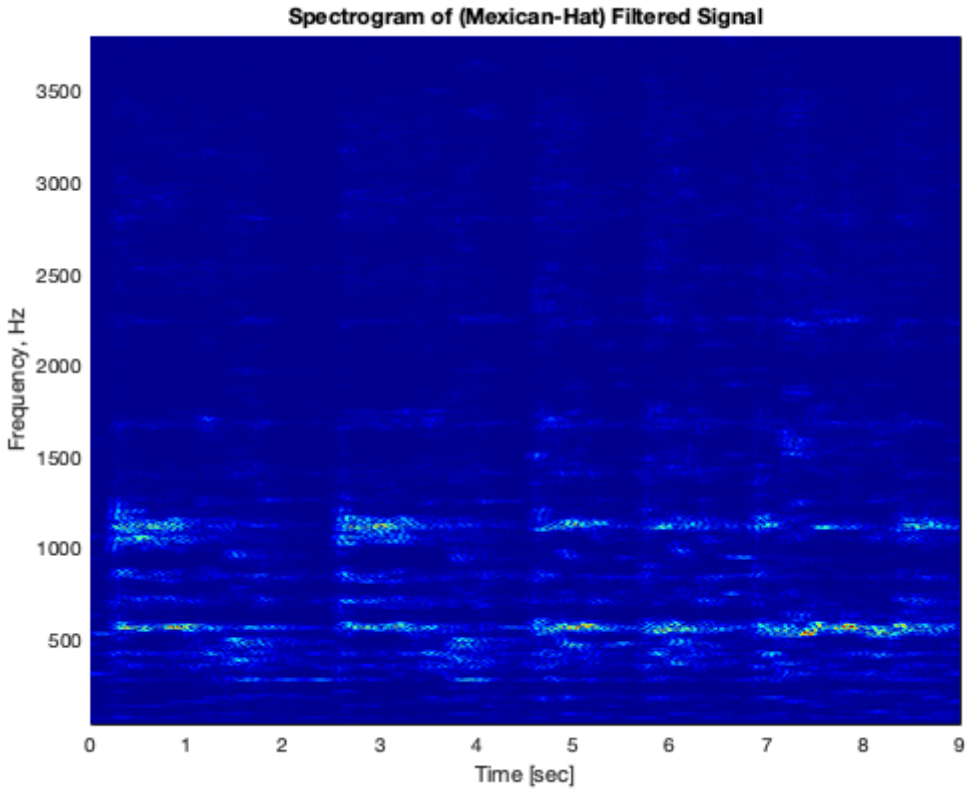

Sliding of Mexican-Hat Filter over Time
Signal(t) & HatFilter(t)
4
2
0
0 1 2 3 4 5 6 7 8 9
Time (t)
Signal(t)*HatFilter(t)
0.5
0
-0.5
0 1 2 3 4 5 6 7 8 9
Time (t)
FFT(Signal*Filter)
1
0.5
0
-3 -2 -1 0 1 2 3
Freq ω
×10^4

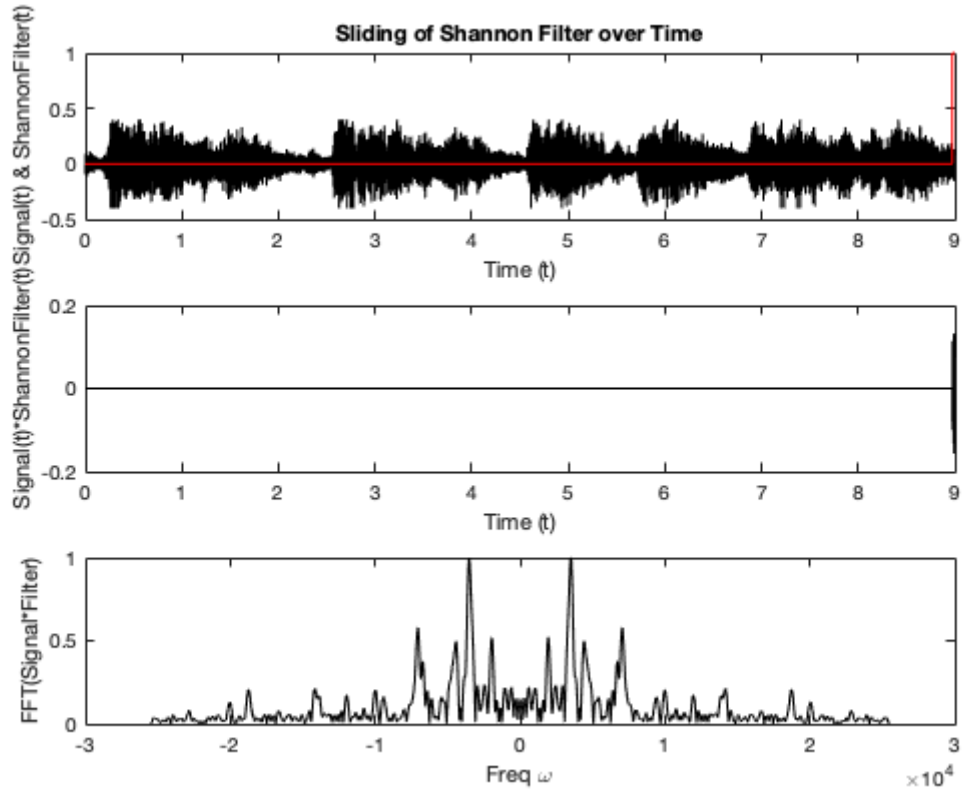


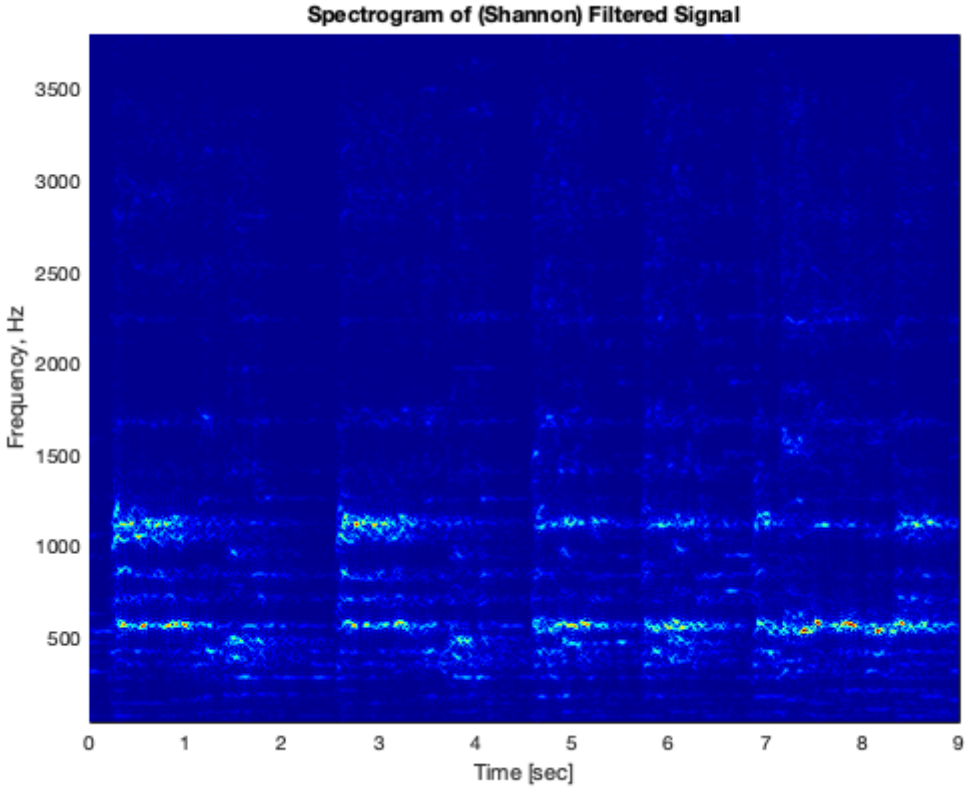
Gabor Transform, Shannon Filter

```
Sgt_spec = [];
for j=1:length(tslide)
    g = abs(t-tslide(j)) <= 0.05/2; % Shannon
    Sg = g .* S;
    Sgt = fft(Sg);
    Sgt_spec = [Sgt_spec; abs(fftshift(Sgt))];

    figure(6)
    subplot(3,1,1)
    plot(t, S, 'k', t, g, 'r')
    xlabel('Time (t)'), ylabel('Signal(t) & ShannonFilter(t)')
    title('Sliding of Shannon Filter over Time')
    subplot(3,1,2)
    plot(t, Sg, 'k')
    xlabel('Time (t)'), ylabel('Signal(t)*ShannonFilter(t)')
    subplot(3,1,3)
    plot(ks, abs(fftshift(Sgt))/max(abs(Sgt)), 'k')
    xlabel('Freq \omega'), ylabel('FFT(Signal*Filter)')

    drawnow
    pause(0.00001)
end
```

```
figure(7)
pcolor(tslide, ks/(2*pi), Sgt_spec.'), shading interp
colormap(jet)
ylim([38 3800])
xlabel('Time [sec]')
ylabel('Frequency, Hz')
title('Spectrogram of (Shannon) Filtered Signal')
```

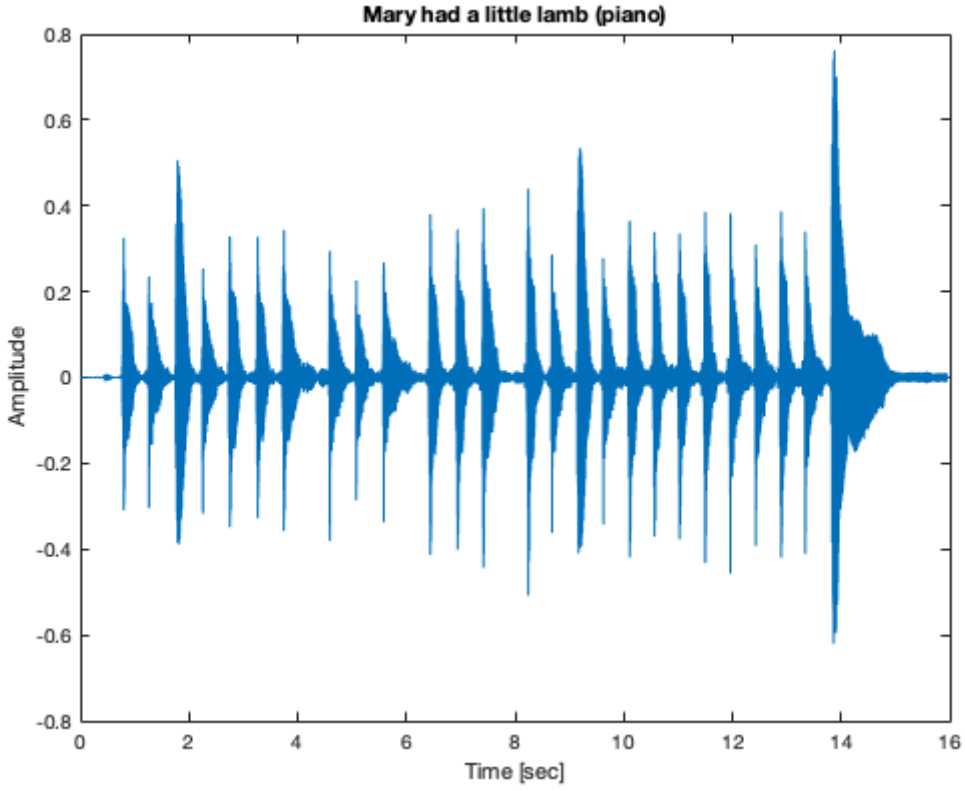


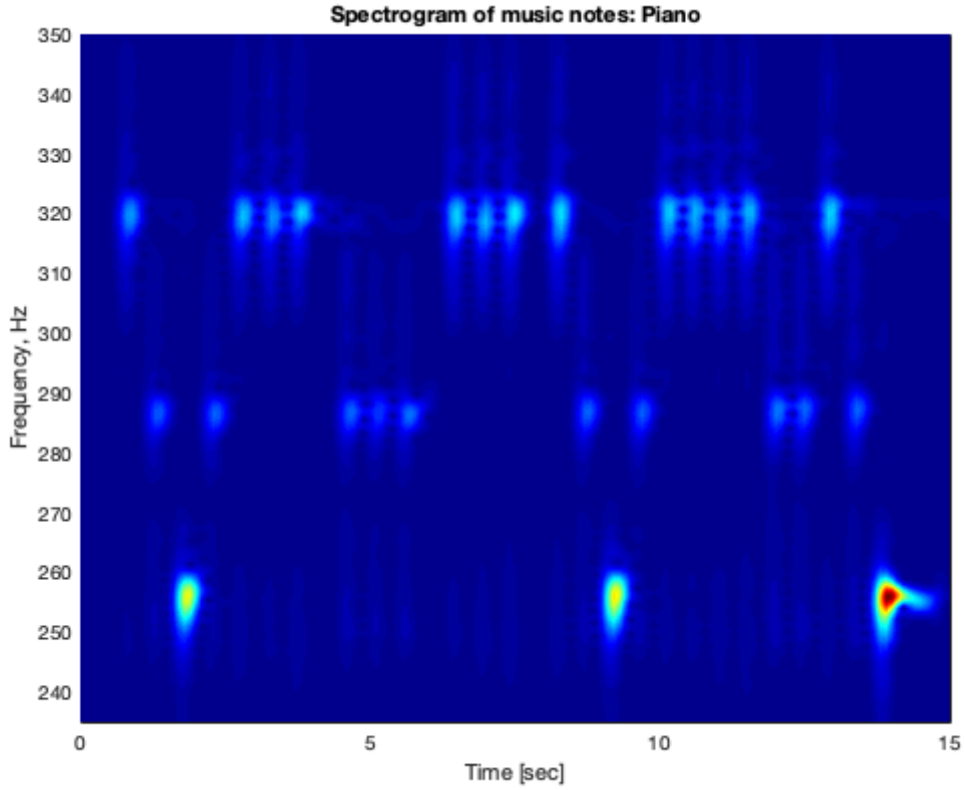

Sliding of Shannon Filter over Time
Signal(t) & ShannonFilter(t)
Time (t)
Signal(t)*ShannonFilter(t)
Time (t)
FFT(Signal*Filter)
Freq ω
×10^4


Spectrogram of (Shannon) Filtered Signal
Frequency, Hz
Time [sec]

Part 2: Mary Had A Little Lamb: Piano

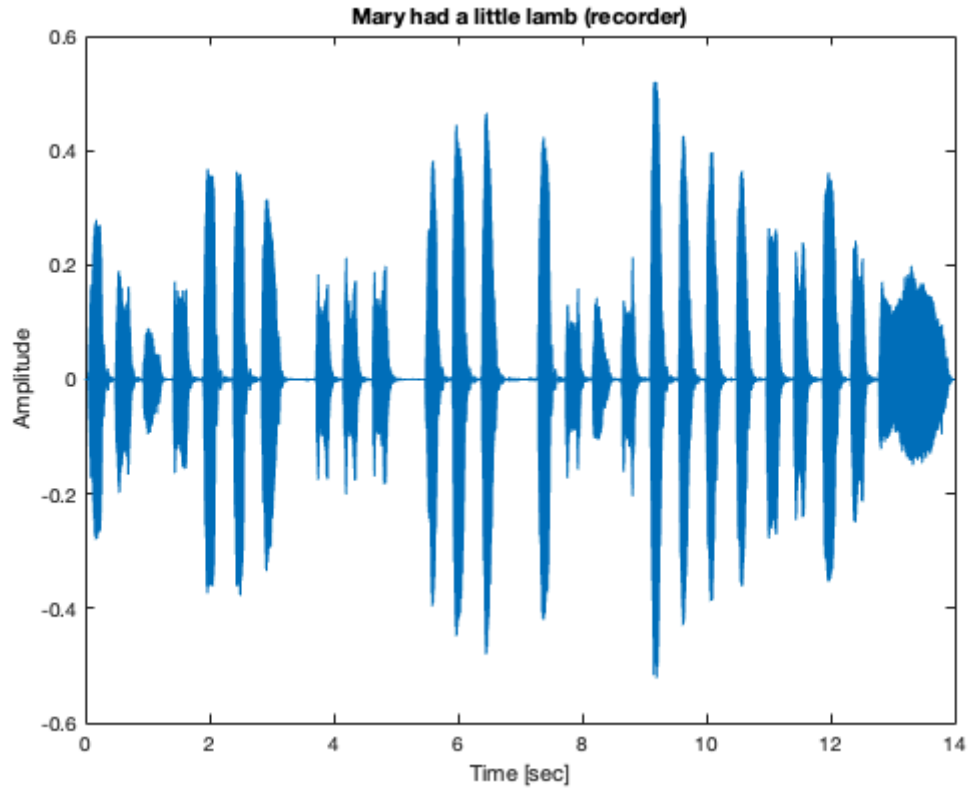
```
tr_piano=16; % record time in seconds
y_piano=audioread('music1.wav');
Fs_piano=length(y_piano)/tr_piano;
S_piano = y_piano';
n=length(S_piano); % Fourier modes
t_piano=linspace(0,tr_piano,n);
k_piano=(2*pi/tr_piano)*[0:n/2-1 -n/2:-1]; % Freq space rescaled to
 2pi domain
ks_piano=fftshift(k_piano); % Unshift k

figure(8)
plot((1:length(y_piano))/Fs_piano,y_piano);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Mary had a little lamb (piano)'); drawnow
%p8 = audioplayer(y,Fs); playblocking(p8);

Sgt_spec_piano = [];
tslide_piano = 0:0.1:15;
for j=1:length(tslide_piano)
    g = exp(-30*(t_piano-tslide_piano(j)).^2); % Gabor
    Sg = g .* S_piano;
    Sgt = fft(Sg);
    Sgt_spec_piano = [Sgt_spec_piano; abs(fftshift(Sgt))];
end
figure(9)
%change y in plot to be Hz.
pcolor(tslide_piano, ks_piano/(2*pi), Sgt_spec_piano.'),
 shading interp
colormap(jet)
title('Spectrogram of music notes: Piano')
ylim([235 350])
xlabel('Time [sec]')
ylabel('Frequency, Hz')
```

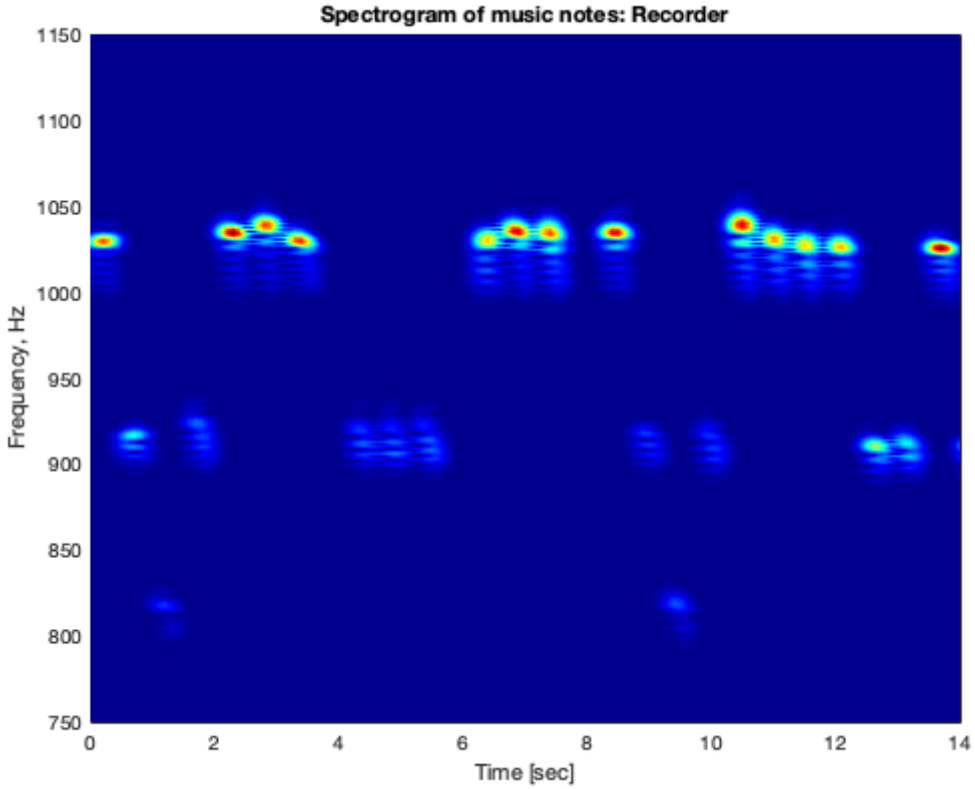

Mary had a little lamb (piano)
Amplitude
Time [sec]
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
2
4
6
8
10
12
14
16


Spectrogram of music notes: Piano
Frequency, Hz
Time [sec]
350
340
330
320
310
300
290
280
270
260
250
240
0
5
10
15

Part 2: Mary Had A Little Lamb: Recorder

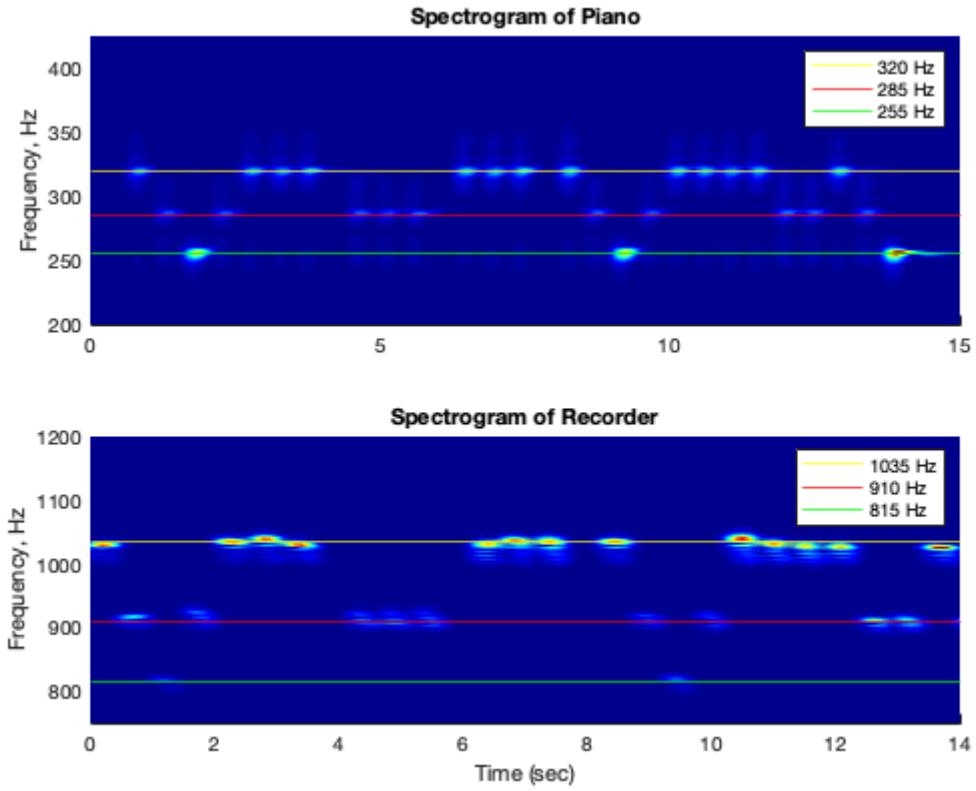
```
tr_rec=14; % record time in seconds
y=audioread('music2.wav');
Fs_rec=length(y)/tr_rec;
S_rec = y';
n=length(S_rec); % Fourier modes
t_rec=linspace(0,tr_piano,n);
k_rec=(2*pi/tr_rec)*[0:n/2-1 -n/2:-1]; % Freq space rescaled to 2pi
 domain
ks_rec=fftshift(k_rec); % Unshift k

figure(10)
plot((1:length(y))/Fs_rec,y);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Mary had a little lamb (recorder)');
%p8 = audioplayer(y,Fs); playblocking(p8);

Sgt_spec_rec = [];
tslide_rec = 0:0.1:14;
for j=1:length(tslide_rec)
    g = exp(-20*(t_rec-tslide_rec(j)).^2); % Gabor
    Sg = g .* S_rec;
    Sgt = fft(Sg);
    Sgt_spec_rec = [Sgt_spec_rec; abs(fftshift(Sgt))];
end
figure(11)
%change y in plot to be Hz.
pcolor(tslide_rec, ks_rec/(2*pi), Sgt_spec_rec.'), shading interp
colormap(jet)
title('Spectrogram of music notes: Recorder')
ylim([750 1150])
xlabel('Time [sec]')
ylabel('Frequency, Hz')
```


Mary had a little lamb (recorder)
Amplitude
Time [sec]
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
2
4
6
8
10
12
14


Spectrogram of music notes: Recorder
Frequency, Hz
Time [sec]
1150
1100
1050
1000
950
900
850
800
750
0
2
4
6
8
10
12
14

Piano and Recorder Spectrograms Together

```
figure(12)

subplot(2,1,1)
y1 = yline(320, 'y');
hold on
y2 = yline(285, 'r');
y3 = yline(255, 'g');
pcolor(tslide_piano, ks_piano/(2*pi), Sgt_spec_piano.'),
 shading interp
colormap(jet)
ylabel('Frequency, Hz')
title('Spectrogram of Piano')
ylim([200 425])
hold off
legend([y1 y2 y3], {'320 Hz','285 Hz','255 Hz'})

subplot(2,1,2)
y4 = yline(1035, 'y');
hold on
y5 = yline(910, 'r');
y6 = yline(815, 'g');
pcolor(tslide_rec, ks_rec/(2*pi), Sgt_spec_rec.'), shading interp
colormap(jet)
xlabel('Time (sec)'), ylabel('Frequency, Hz')
title('Spectrogram of Recorder')
ylim([750 1200])
hold off
legend([y4 y5 y6], {'1035 Hz', '910 Hz', '815 Hz'});
```



Published with MATLAB® R2018b