Time Frequency Analysis of Music Using the Gabor Tranfsorm

Rachel Carroll

02/16/2020

Abstract

This paper investigates time frequency analysis of musical pieces using Gabor filtering and the Fourier transform. In this method, the Fourier transform is taken on intervals centered at discrete points in time to obtain an approximation of the frequency content over time. First, we experiment with various sampling rates and Gabor filter types to identify how these factors affect the time and frequency detail achieved. Then, we apply this method to recordings of "Mary had a little lamb" to find the score of this piece of music.

1 Introduction and Overview

This paper uses the Gabor transform to perform time frequency analyses on samples of musical pieces. In this method, Gabor filters are used to isolate samples of sound wave data at discrete points in time. Then the Fourier transform is applied in order to obtain the frequency content on the specified time intervals. In repeating this process across the entire timeframe of the musical piece, we construct a time and frequency map of the song (aka the spectrogram).

Due to the Heisenberg Uncertainty Principal (described in the following section), this method has an inherent tradeoff between time domain and frequency detail. In many applications, including music, it is especially important to know both time and frequency of a signal since these attributes give us the beat and notes of the music. In this paper we experiment with sampling rates, filters in an attempt to optimize the time and frequency detail balance. In Part 1, Handel's Messiah is analyzed with such methods. In Part 2 we apply the methods to recordings of "Mary had a little lamb" to construct the score for both the piano and recorder.

2 Theoretical Background

Limitations of the Fourier Transform in Time Series Analysis: The Fourier transform is a fundamental method for capturing the frequency of a signal in time or space. When the Fourier transform is taken on a signal in time, all frequencies throughout the time period are captured, however the original time domain information is lost in the process. This a significant issue when we are looking at a signal that is dependent on time, such as music.

Heisenberg Uncertainty Principal: In Quantum mechanics the Heisenberg Uncertainty Principal states that the momentum and position of a partial cannot be known simultaneously. This is a property of the Fourier transform which is why it cannot be used to identify both the time and frequency of a signal simultaneously.

Gabor Transform: The Gabor transform is a version of the Fourier transform that incorporates an additional function of specified width and time so that instead of taking the Fourier transform over the entire time domain (and losing all time information), it takes the Fourier transform on a window

of some width centered at time t . The Gabor transform is represented by

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

Where $g(\tau - t)$ localizes the Fourier integral around $t = \tau$. The width and shape of the Gabor transform depend on how the function g is defined. Because the Gabor measures frequencies at specified points in time, we now have a time frequency analysis method the preserves both time and frequency information. You may be thinking "wow! time AND frequency all in one? This solves everything!" But not so fast....there is a catch.

Limitations of the Gabor Transform: Even though the Gabor transform method preserves some time information, it must sacrifice some frequency content to do so. A measurement highly accurate in time will have a poor frequency results, and vice versa. A narrow Gabor filter window will have great time accuracy because it isolates a small window of time for each measurement. However, the low frequencies will be lost in this case because low frequencies have long wave numbers and therefore will only be captured in a wide window. On the other hand, a wide filter will increase the frequency reading but limit the time detail. This trade off will be explored in this paper. We will experiment with various Gabor filtering methods in an attempt to find the best of both worlds.

3 Algorithm Implementation and Development

In this analysis MATLAB was used to read in the sound data for Handel's Messiah and the recordings of "Mary Had a Little Lamb" beautifully played by a budding musician on both the piano and recorder. This analysis was split into two parts, one for each song.

3.1 Part 1: Analysis of Handel

The following steps were taken to identify the effects of different Gabor filters and sampling rates on a sample of Handel's Messiah.

1. **Load in music and set up time and frequency domains:** First the music sound data was read into MATLAB. This came in the form of a vector of amplitudes (y) and a given sampling frequency (Fs). The time domain is a vector of equally incremented values from 0 to around 9 seconds discretized with the same number of points as the amplitude vector. Since we plotted frequency in Hertz, the Fourier domain the sampling frequency centered at zero (from $-\frac{Fs-1}{2}$ to $\frac{Fs-1}{2}$) where the number of nodes also matches the length of the music vector. For this piece of music the sampling frequency is $73,113$. The bounds of the Fourier domain are defined as they are so that each end is an integer.

2. **Effects of Gaussian filters of varying widths:** Three Gaussian filters with widths of 0.5, 2, and 50 were used to demonstrate the effects of different filter widths. The music was sampled every 0.1 seconds.

3. **Effects of Gaussian filters of fixed width at varying sampling rates:** In exploring the sampling rate effects a Gaussian filter was used again but this time with a relatively narrow fixed width of 20. Two spectrographs were created, one where the music was samples every half a second and one where it was samples every .01 seconds.

4. **Effects of filter shapes using similar widths and sampling rates:** The results from a Gaussian, Mexican Hat, and Shannon Step filter were compared. Each filter sampled at .01 seconds and had similar widths.

3.2 Part 2: Analysis of Mary Had Little Lamb on Piano and recorder

The following steps were taken to analyze both the piano and recorder music separately.

1. **Read in music and define domains:** This step is the same as above, the only difference is that the frequency domain is from $-\frac{Fs}{2}$ to $\frac{Fs}{2}$ since the sampling frequency is even, so no need to subtract by 1.

2. **Plot overall time and frequency footprints:** Luckily, "Mary had a little lamb" is a simple song with only three notes. Therefore we can get relevant information about the recording by taking the Fourier transform of the entire piece. By taking the overall Fourier transform, the three most relevant frequencies were identified for each recording.

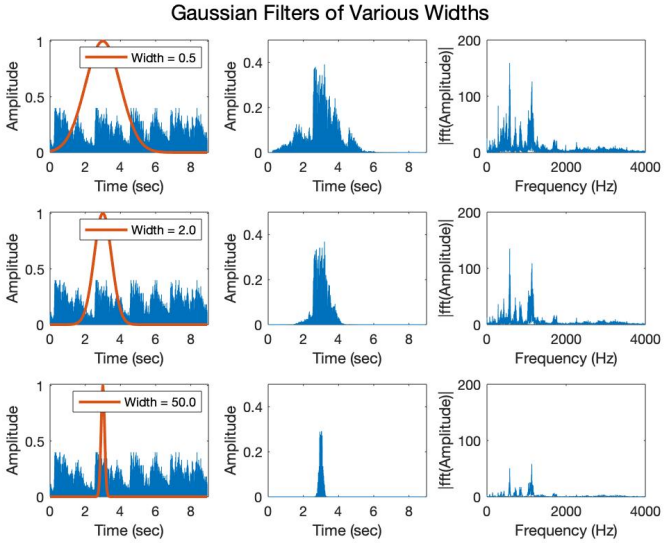
3. **Create spectrograms of each recording:** The spectrogram of each recording were created with gaussian filters of width 20 and sampling rate of 0.1 seconds.

4. **Analyze score of each recording:** Finding the notes analogous to the identified frequencies, and using the spectrogram for the timing, the score of the music was constructed.

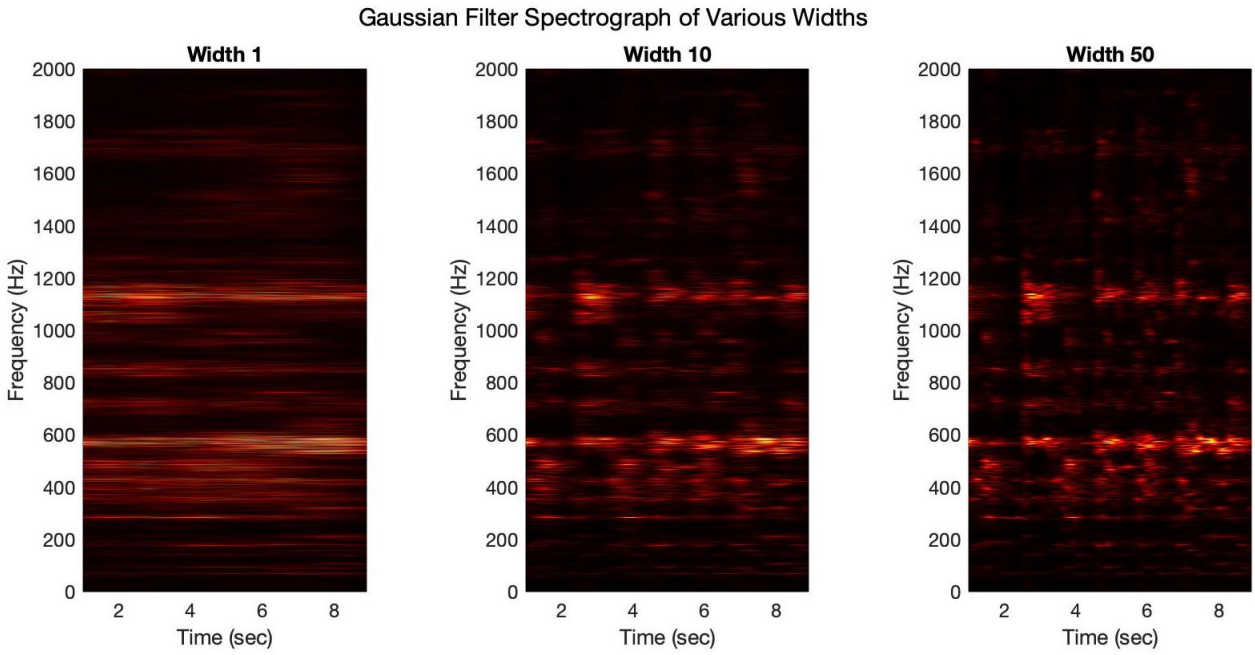
4 Computational Results

4.1 Part 1

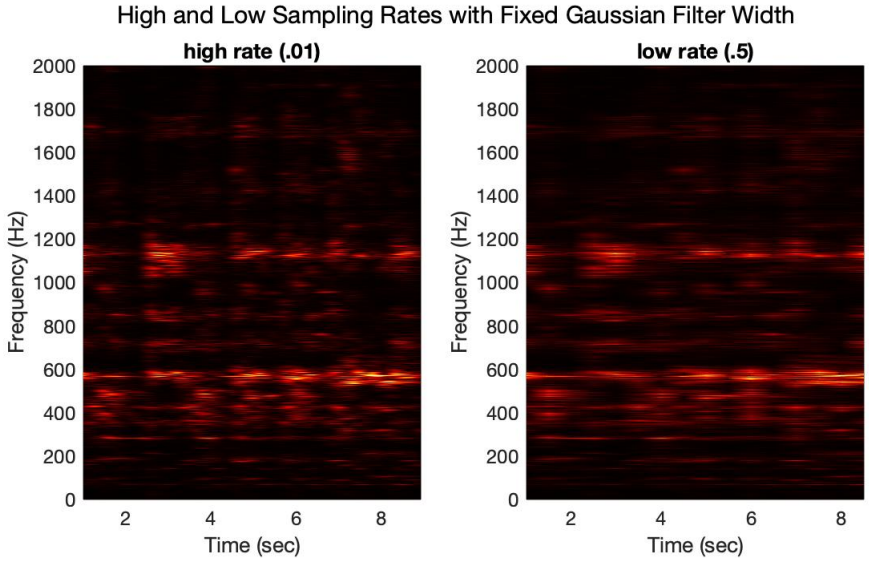
Gaussian Filter Widths: The figures below show that a wide filter yields high frequency content but loses a lot of time information. On the other hand, as width narrows, the point in time is very specific but much of a the frequency content is lost.



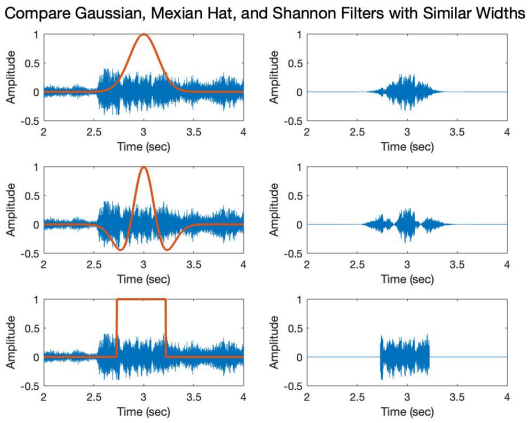
The spectrogram upholds this assessment. Further, it shows that even with a set sampling rate, the width makes a big different in the quality of the time and frequency content achieved.



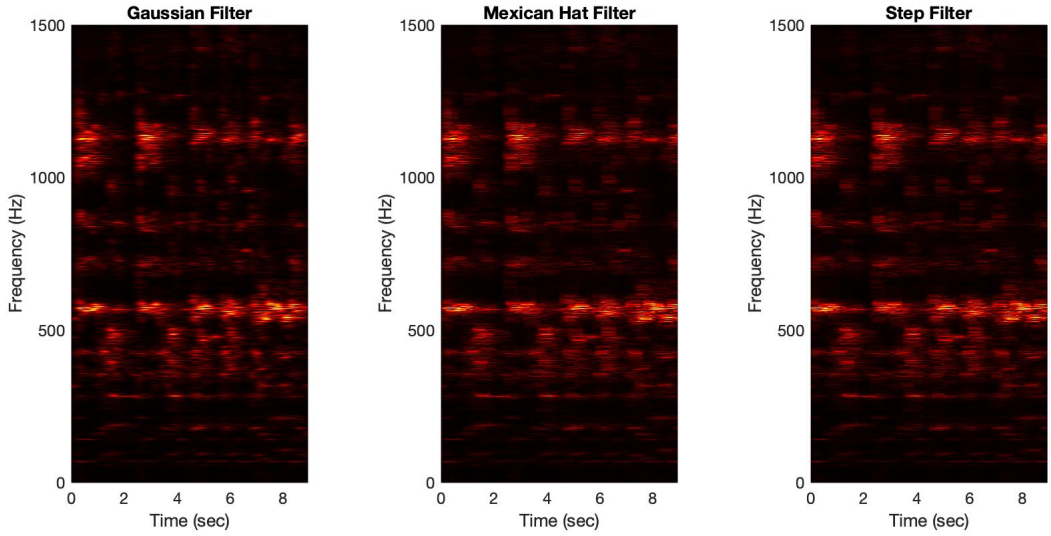
Over/Under Sampling: The converse of above, the image below shows how with a fixed filter, the sampling rate makes a big difference. Notice how the low sample rate gives poor time content despite the fact that a relatively narrow filter was used.



Gabor Filter Shapes: The figure below shows the three Gabor filter shapes used. As shown, they have similar widths, but they filter the signal quite differently.

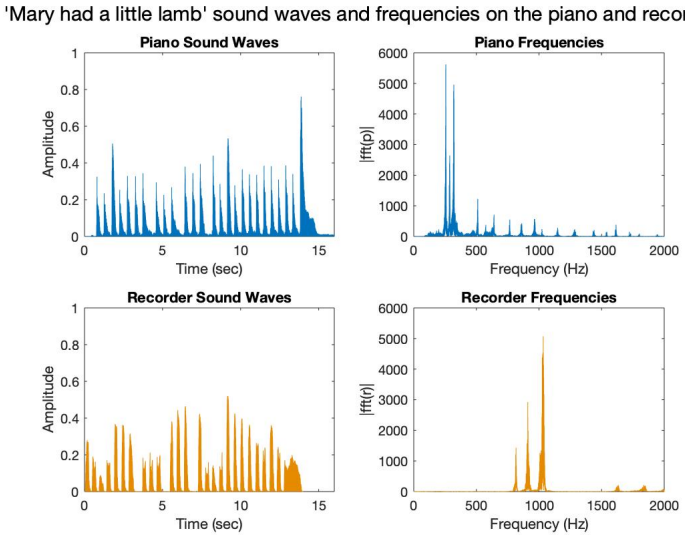


The differences in shape are more subtle than in changing widths and sampling rates in this case. However, the Step and Mexican hat filter out perform the Gaussian in frequency detail. Although its slight, even little changes can go a long was when working with such large data sets.

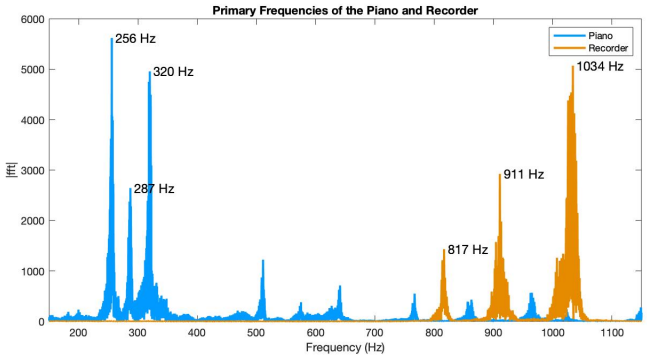


4.2 Part 2

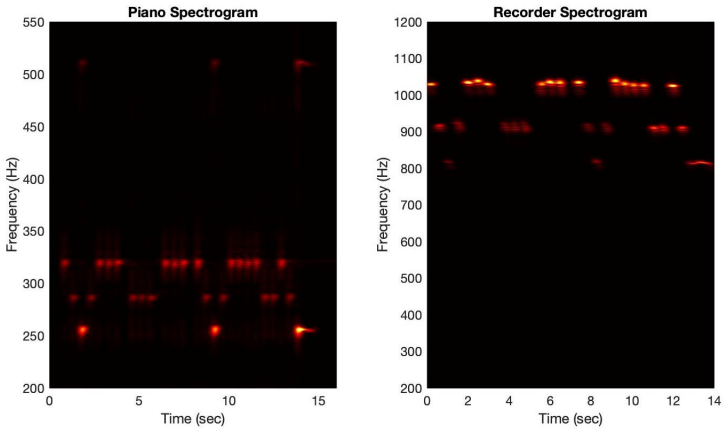
Overall Frequency: The figure below shows the sound wave and frequency content of "Mary had a little lamb" on the piano and recorder. Notice that for each recording, there are three spikes in the frequency graph at the three prominent frequencies. These represent the three notes of the song. Of course, from these plots we cannot tell when the notes will occur in the song, but it gives us a good idea of what frequencies we expect to see when we analyze the recordings in time
The frequency plots also show us that overall the piano is playing lower frequencies than the recorder. Another clear qualitative difference between the pieces is the existence of overtones. Instruments can generate overtones at integer multiples of the central frequency. The overtones generated by an instrument has a significant effect on the quality of its sound. In musical jargon, the existence of overtone is referred to as "timbre". As the frequency plots show, the piano generated far more overtones than the recorder as seen by the smaller spikes in frequency content beyond the primary three frequencies.



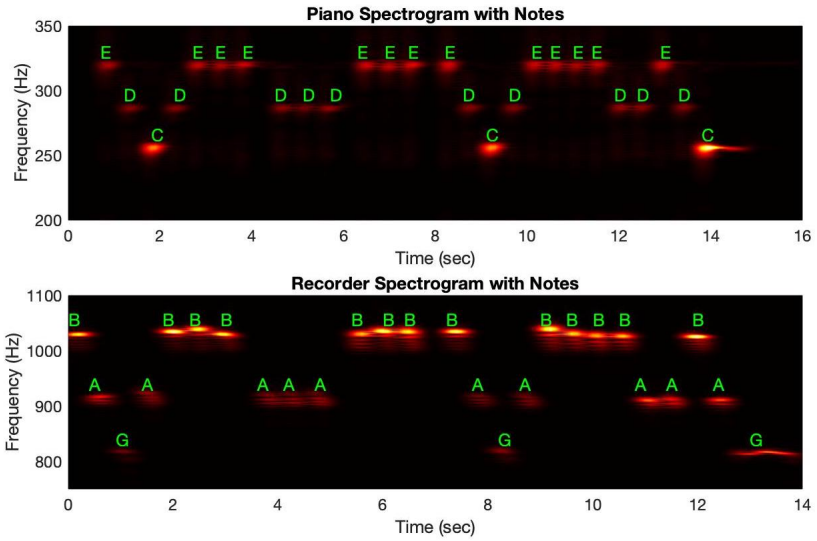
Prominent Frequencies: Below is a direct comparison the the prominent frequencies of the piano and recorder. The prominent frequencies are noted. These values are used to identify which note is being played.



Music Score Creation: The final piece of the puzzle is the time content. Below we can see from the spectrogram and pretty clear map of the recordings in both time and frequency.



Putting all the pieces together (and using a map between frequency and notes) we have the following "score" of each piece



5 Summary and Conclusions

This project demonstrates both the power and limitations of the Gabor transform methodology on time series signals. On the one hand, the Heisenberg Uncertainty Principal limits our ability to fully capture exact frequency and time domain information from a signal. However, by experiementing with different filter sizes and shapes, we can extract representations of a signal with both time and frequency content. This type of work is as much an art as it is a science and it also greatly depends on the signal you are working with and the overall goal of the analysis.

6 Appendix A MATLAB functions used and brief implementation explanation

The following functions were important in this analysis:

- fft() - computes the fourier transform for an n dimensional array of data
- fftshift() - shifts the domain to its original order because, to increase computational efficiency, the fftn() function shifts the domain so that $[-L, 0] \to [0, L]$ and $[0, L] \to [-L, 0]$.
- pcolor - pseudocolor plot of matrices. Used to plot spectrographs to highlight frequencies content in discrete points in time.
- colormap - dictates colors used in pcolor. in this analysis we went with "hot" which varies smoothly from black, through shades of red, orange, and yellow, to white
- audioread - reads in audio (.wav) file. Used to read in the pieano and recorder vectors of "mary had a little lamb"

7 Appendix B MATLAB Code

7.1 Part1

Table of Contents

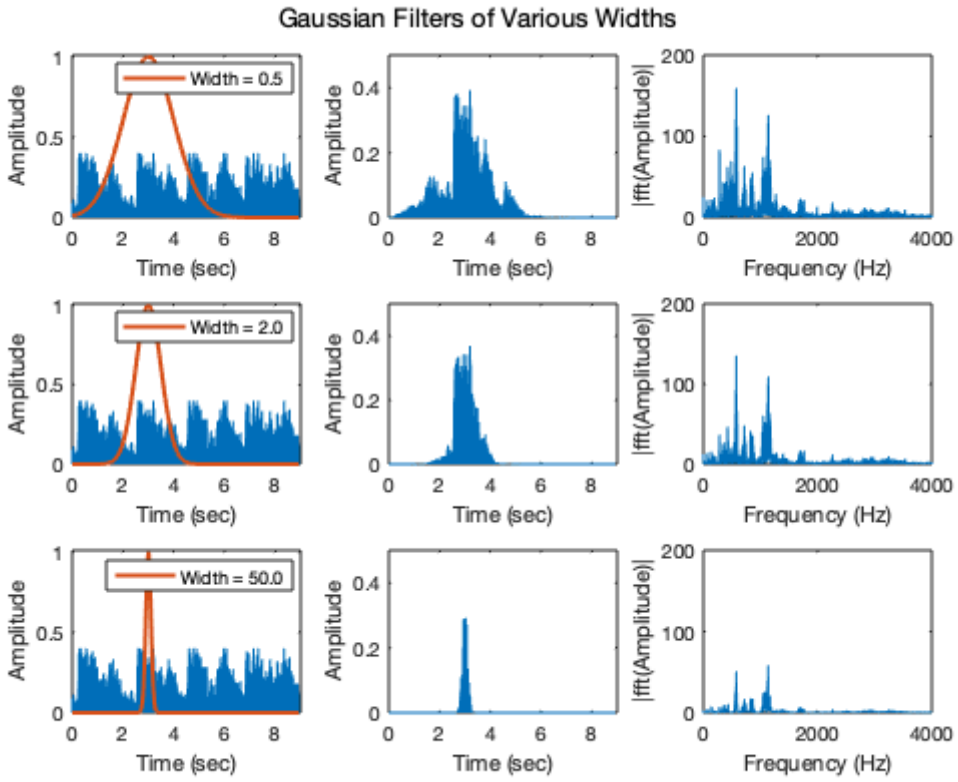
Load in Handel Music and Set up Time and Frequency Domains

```
clear all; close all; clc

load handel
v = y'/2;
%Set up domains
L=length(v)/Fs; n=length(v);
t2=linspace(0,L,n+1); t=t2(1:n); %time domain
ks=(1/L)*[-(n-1)/2:(n-1)/2];   %freq domain
```

View at Gausian Filter Options

```
figure(1)
width = [.5, 2, 50]; %Gaussian width options
t_center=3;
for i=1:length(width)
    g=exp(-width(i)*(t-t_center).^2); %Gausian filter centered at t=3
    v_filt = g.*v;
    v_filt_transform = fft(v_filt);
    vt_plot = fftshift(abs(v_filt_transform));

    subplot(length(width),3,3*i-2) %plot signal with Gaussian on top
    p1=plot(t,v); hold on
    p2=plot(t,g,'Linewidth',[2]);
    legend([p2],sprintf('Width = %.1f', width(i)));
    axis([0 9 0 1.01]);
    xlabel('Time (sec)');
    ylabel('Amplitude');

    subplot(length(width),3,3*i-1); %plot filtered signal
    plot(t,v_filt);
    xlabel('Time (sec)');
    ylabel('Amplitude');
    axis([0 9 0 .5])

    subplot(length(width),3,3*i); %plot frequency content of filtered
 signal
    plot(ks,vt_plot);
    xlabel('Frequency (Hz)');
```

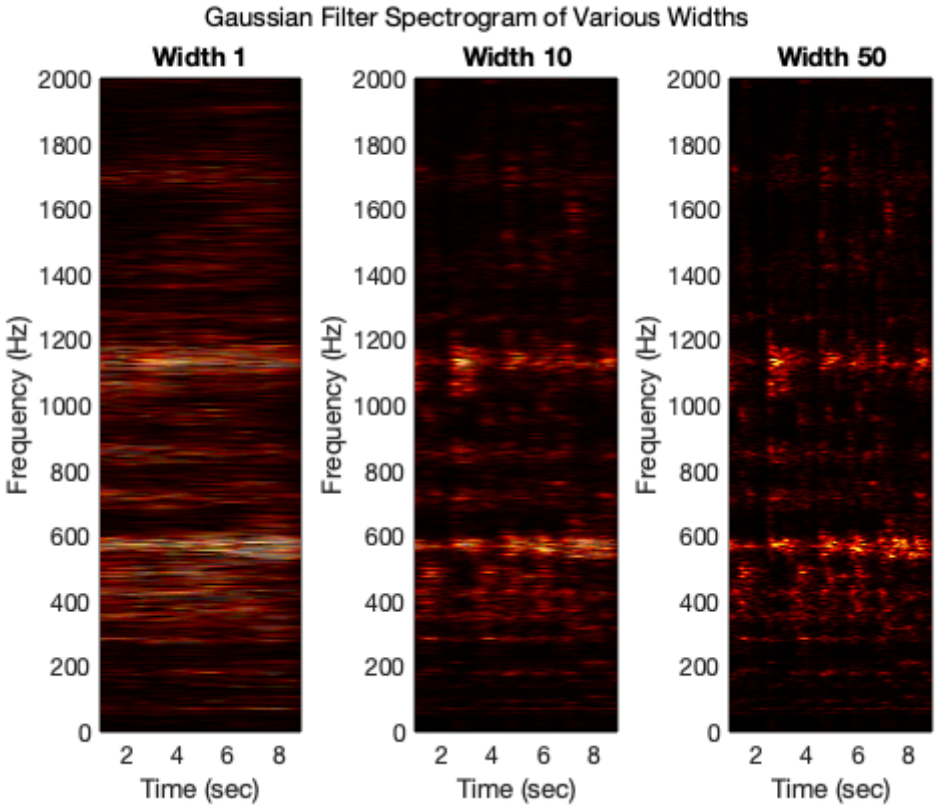
```
    ylabel('|fft(Amplitude)|');
    axis([0 4000 0 200]);
end

sgtitle('Gaussian Filters of Various Widths');
```



Spectrograms with Different Gaussian Filter Widths

```
figure(2)
width = [1,10,50];
for j=1:length(width)
    t_center=[1:.1:L];
    spec_gw=[];
    for i=1:length(t_center)
        g=exp(-width(j)*(t-t_center(i)).^2);
        v_filt = g.*v;
        v_filt_transform = fft(v_filt);
        vt_plot = fftshift(abs(v_filt_transform));
        spec_gw=[spec_gw;vt_plot];
    end

subplot(1,length(width),j)
pcolor(t_center,ks,spec_gw.'),
shading interp
```

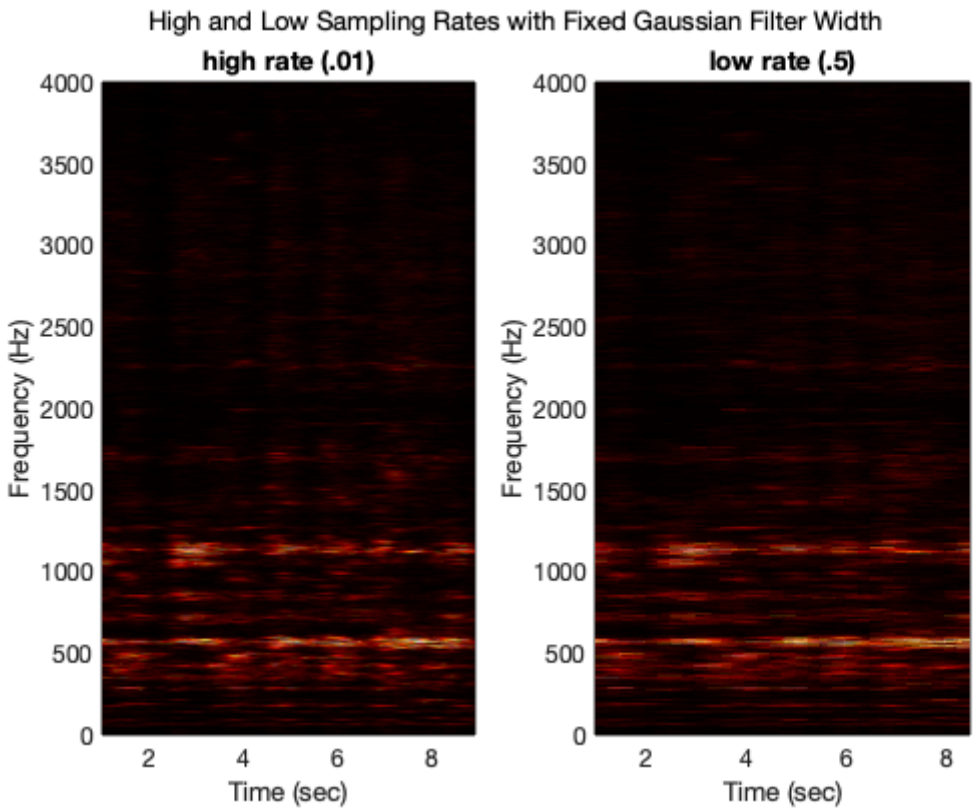
```
set(gca,'Ylim',[0 2000],'Fontsize',[12])
xlabel('Time (sec)');
ylabel('Frequency (Hz)');
title(strcat("Width ",num2str(width(j))))
colormap(hot)
end

sgtitle("Gaussian Filter Spectrogram of Various Widths")
```



Hold Gaussian Filter Width Constant to Evaluate Over/Under sampling effects

```
width = 20;

% Low sampling
t_center_1=[1:.5:L];
spec_g_us=[];
for i=1:length(t_center_1)
    g=.4*exp(-width*(t-t_center_1(i)).^2);
    v_filt = g.*v;
    v_filt_transform = fft(v_filt);
    vt_plot = fftshift(abs(v_filt_transform));
    spec_g_us=[spec_g_us;vt_plot];
end
```

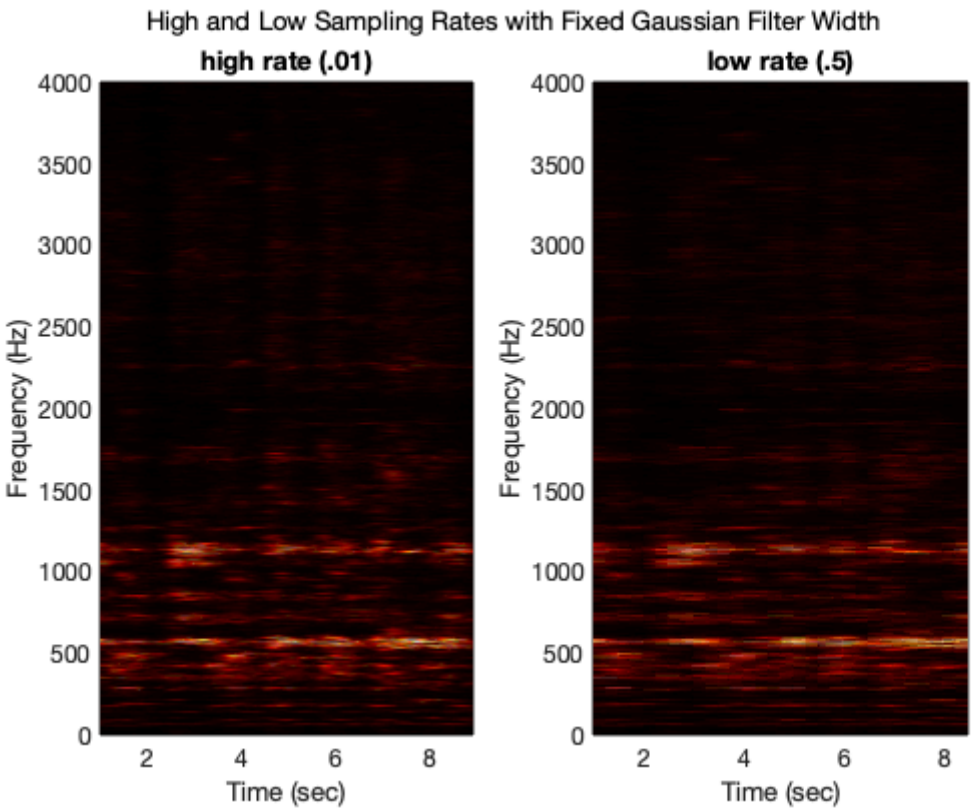
```
% High sampling
t_center=[1:.01:L];
spec_g_os=[];
for i=1:length(t_center)
    g=.4*exp(-width*(t-t_center(i)).^2);
    v_filt = g.*v;
    v_filt_transform = fft(v_filt);
    vt_plot = fftshift(abs(v_filt_transform));
    spec_g_os=[spec_g_os;vt_plot];
end

%Display Spectrogram
figure(3)

subplot(1,2,1)
pcolor(t_center,ks,spec_g_os.'),
shading interp
set(gca,'Ylim',[0 4000],'Fontsize',[12])
colormap(hot)
xlabel('Time (sec)');
ylabel('Frequency (Hz)');
title("high rate (.01)")

subplot(1,2,2)
pcolor(t_center_1,ks,spec_g_us.'),
shading interp
set(gca,'Ylim',[0 4000],'Fontsize',[12])
colormap(hot)
xlabel('Time (sec)');
ylabel('Frequency (Hz)');
title("low rate (.5)")

sgtitle("High and Low Sampling Rates with Fixed Gaussian Filter
 Width")
```

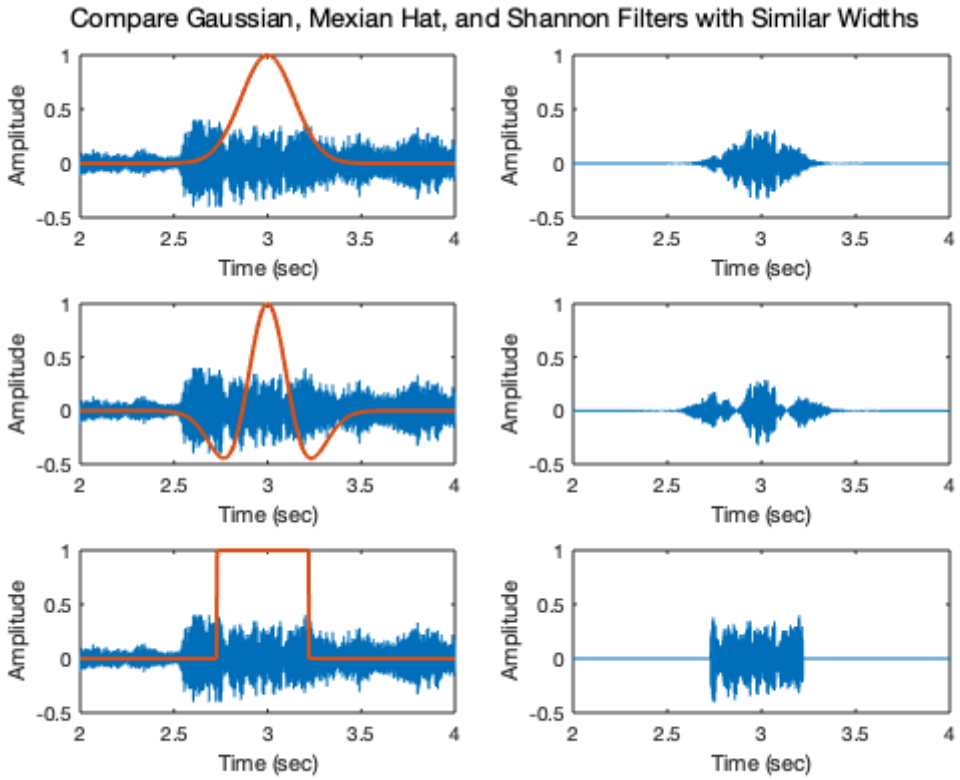


Compare Different Filter Shapes with Similar Widths

```
g_width = 25;
mh_width = 55;
sf_width = 4013;
g_mh_t_center = 3;
g=exp(-g_width*(t-g_mh_t_center).^2); %Gausian filter centered at t=3
mh=(1-mh_width.*(t-g_mh_t_center).^2).*exp((-mh_width.*(t-
g_mh_t_center).^2)./2);
mask=ones(1,sf_width);
sf=zeros(1,n);
tstep=22371;
sf(1+tstep:sf_width+tstep)=mask;


figure(4)
%Gaussian filter plots
subplot(3,2,1)
plot(t,v), hold on
plot(t,g,'Linewidth',[2])
axis([2,4,-.5,1])
xlabel('Time (sec)');
ylabel('Amplitude');
```

```
subplot(3,2,2)
plot(t,g.*v), hold on
axis([2,4,-.5,1])
xlabel('Time (sec)');
ylabel('Amplitude');

%Mexican Hat filter plots
subplot(3,2,3)
plot(t,v), hold on
plot(t,mh,'Linewidth',[2])
axis([2,4,-.5,1])
xlabel('Time (sec)');
ylabel('Amplitude');

subplot(3,2,4)
plot(t,mh.*v), hold on
axis([2,4,-.5,1])
xlabel('Time (sec)');
ylabel('Amplitude');

%Shannon filter plots
subplot(3,2,5)
plot(t,v), hold on
plot(t,sf,'Linewidth',[2])
axis([2,4,-.5,1])
xlabel('Time (sec)');
ylabel('Amplitude');

subplot(3,2,6)
plot(t,sf.*v), hold on
axis([2,4,-.5,1])
xlabel('Time (sec)');
ylabel('Amplitude');

sgtitle("Compare Gaussian, Mexian Hat, and Shannon Filters with
 Similar Widths")
```

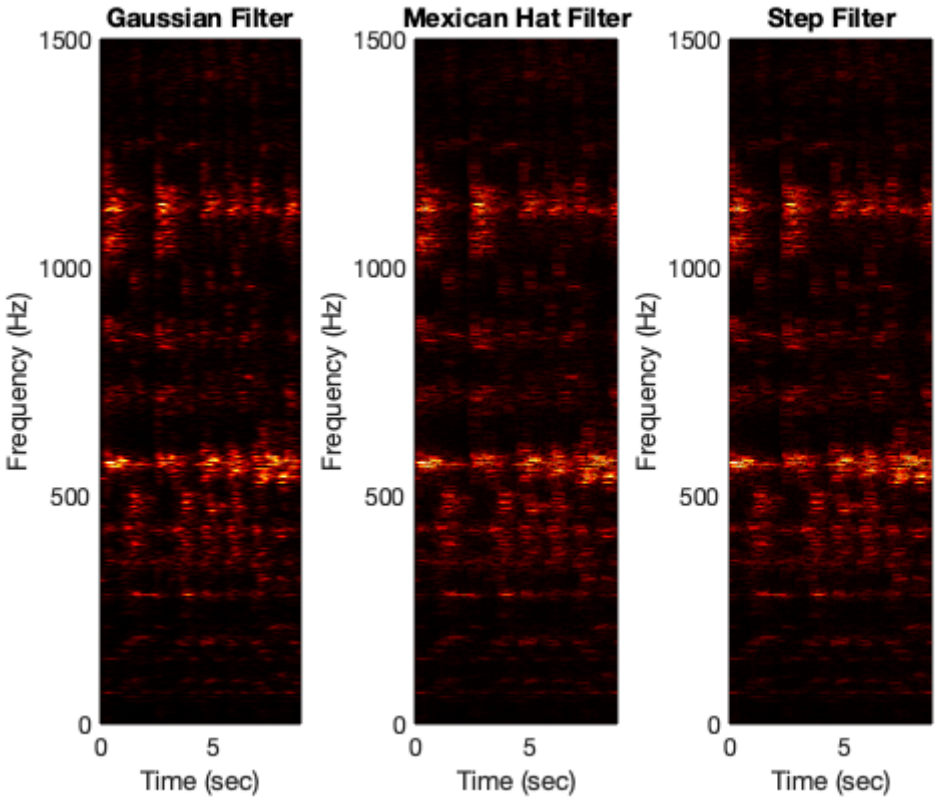

High and Low Sampling Rates with Fixed Gaussian Filter Width
high rate (.01)
low rate (.5)
Frequency (Hz)
Time (sec)
4000
3500
3000
2500
2000
1500
1000
500
0
2
4
6
8


Compare Gaussian, Mexian Hat, and Shannon Filters with Similar Widths
Amplitude
Time (sec)
1
0.5
0
-0.5
2
2.5
3
3.5
4

Spectrograms of different filter shapes with same width and sampling rate

```
g_width = 25;
mh_width = 55;
sf_width = 4013;
t_center = linspace(0,L,80);

steps=length(t_center);
width_approx=sf_width;
tstep_mult=round((73113-width_approx)/steps);
calc_width=73113-(tstep_mult*steps);
sf=zeros(1,n);
mask=ones(1,calc_width);

spec_sf=[];
spec_g=[];
spec_mh=[];

for i=1:length(t_center)
    g=exp(-g_width*(t-t_center(i)).^2);
    v_filt_g = g.*v;
    v_filt_g_transform = fft(v_filt_g);
    vt_plot_g = fftshift(abs(v_filt_g_transform));
    spec_g=[spec_g;vt_plot_g];

    mh=(1-mh_width.*(t-t_center(i)).^2).*exp((-mh_width.*(t-
t_center(i)).^2)./2);
    v_filt_mh = mh.*v;
    v_filt_mh_transform = fft(v_filt_mh);
    vt_plot_mh = fftshift(abs(v_filt_mh_transform));
    spec_mh=[spec_mh;vt_plot];

    sf=zeros(1,n);
    tstep=(i-1)*tstep_mult;
    sf(1+tstep:calc_width+tstep)=0.41.*mask;
    v_filt_sf = sf.*v;
    v_filt_sf_transform = fft(v_filt_sf);
    vt_plot = fftshift(abs(v_filt_sf_transform));
    spec_sf=[spec_sf;vt_plot];
end

figure(5)

subplot(1,3,1)
pcolor(t_center,ks,spec_g.'),
shading interp
set(gca,'Ylim',[0 1500],'Fontsize',[12])
colormap(hot)
xlabel('Time (sec)');
ylabel('Frequency (Hz)');
title("Gaussian Filter")
```

```
subplot(1,3,2)
pcolor(t_center,ks,spec_mh.'),
shading interp
set(gca,'Ylim',[0 1500],'Fontsize',[12])
colormap(hot)
xlabel('Time (sec)');
ylabel('Frequency (Hz)');
title("Mexican Hat Filter")


subplot(1,3,3)
pcolor(t_center,ks,spec_sf.'),
shading interp
set(gca,'Ylim',[0 1500],'Fontsize',[12])
colormap(hot)
xlabel('Time (sec)');
ylabel('Frequency (Hz)');
title("Step Filter")
```



Published with MATLAB® R2019b

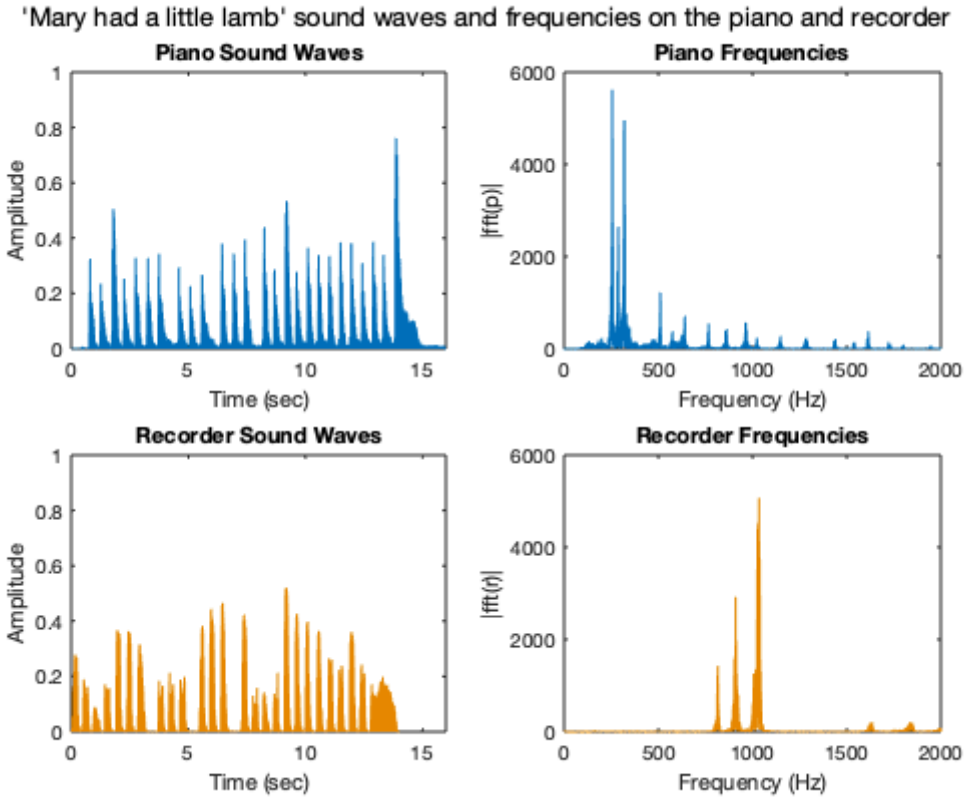
7.2 Part 2

Table of Contents

Read in Music and Plot in Time and Freq Domains

```
clear all; close all; clc

% load piano music
tr_piano=16;  % record time in seconds
p=audioread('music1.wav'); Fs_p=length(p)/tr_piano;
p=p';
L_p=length(p)/Fs_p; n_p=length(p);
ks_p=(1/L_p)*[-n_p/2:(n_p/2)-1]; %frequency domain
t_p=(1:length(p))/Fs_p; %time domain
ftp=fft(p); % fourier transform of piano music
ftp=abs(fftshift(ftp));

%load recorder music
tr_rec=14;  % record time in seconds
r=audioread('music2.wav'); Fs_r=length(r)/tr_rec;
r=r';
L_r=length(r)/Fs_r; n_r=length(r);
ks_r=(1/L_r)*[-n_r/2:(n_r/2)-1];
t_r=(1:length(r))/Fs_r;
ftr=fft(r); % fourier transform of recorder music (overall)
ftr=abs(fftshift(ftr));

%plot overall frequencies and sound waves
figure(1)

subplot(2,2,1)
plot(t_p,p);
xlabel('Time (sec)'); ylabel('Amplitude');
title('Piano Sound Waves');
axis([0,16,0,1])

subplot(2,2,2)
plot(ks_p,ftp);
xlabel('Frequency (Hz)'); ylabel('|fft(p)|');
title('Piano Frequencies');
axis([0,2000,0,6000])

subplot(2,2,3)
```

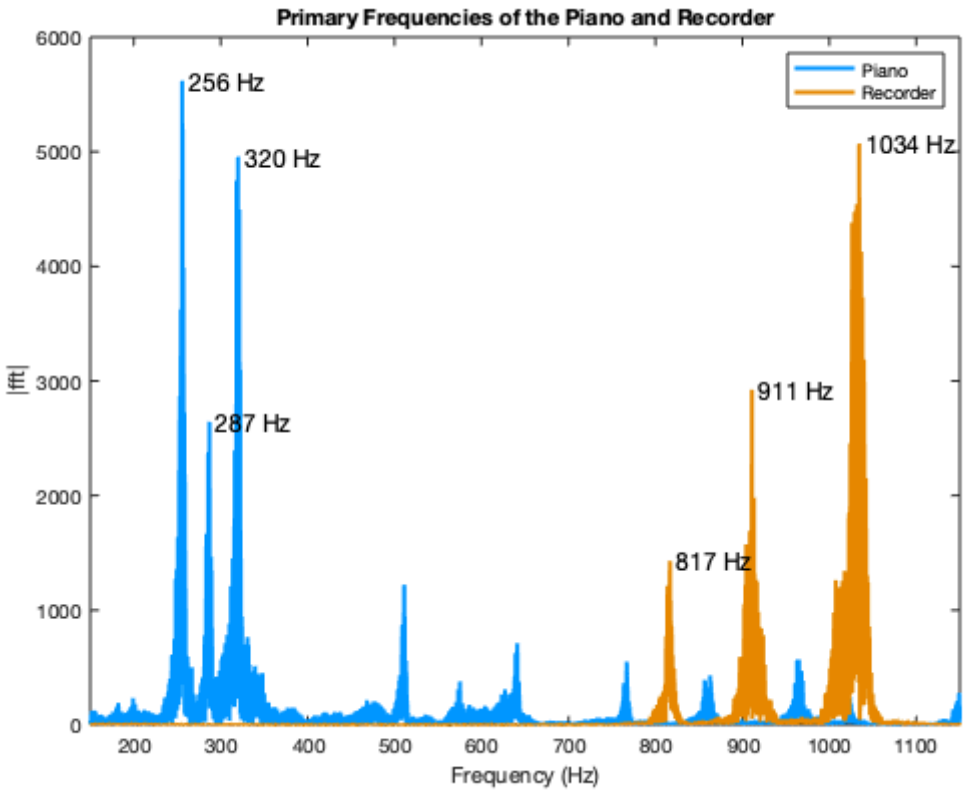
```
plot(t_r,r,'Color','#E68A00');
xlabel('Time (sec)'); ylabel('Amplitude');
title('Recorder Sound Waves');
axis([0,16,0,1])

subplot(2,2,4)
plot(ks_r,ftr,'Color','#E68A00');
xlabel('Frequency (Hz)'); ylabel('|fft(r)|');
title('Recorder Frequencies');
axis([0,2000,0,6000])

sgtitle("'Mary had a little lamb' sound waves and frequencies on the
piano and recorder")
```



Identify the Three Primary Frequencies for Each Instrument

```
%piano
p_note1_max=max(ftp(find(ks_p>240 & ks_p<270)));
p_note2_max=max(ftp(find(ks_p>280 & ks_p<300)));
p_note3_max=max(ftp(find(ks_p>300 & ks_p<350)));
p_note1_freq=ks_p(find(ftp == p_note1_max ,1,'last'));
p_note2_freq=ks_p(find(ftp == p_note2_max ,1,'last'));
p_note3_freq=ks_p(find(ftp == p_note3_max ,1,'last'));
%recorder
```

```
r_note1_max = max(ftr(find(ks_r>800 & ks_r<850)));
r_note2_max = max(ftr(find(ks_r>890 & ks_r<950)));
r_note3_max = max(ftr(find(ks_r>1000 & ks_r<1100)));
r_note1_freq=ks_r(find(ftr == r_note1_max,1,'last'));
r_note2_freq=ks_r(find(ftr == r_note2_max,1,'last'));
r_note3_freq=ks_r(find(ftr == r_note3_max,1,'last'));
%vector of relevant frequencies for plot annotations
note_freq =[p_note1_freq p_note2_freq p_note3_freq r_note1_freq
 r_note2_freq r_note3_freq];
note_max = [p_note1_max p_note2_max p_note3_max r_note1_max
 r_note2_max r_note3_max];
note_txt=strcat(" ",string(round(note_freq))',repmat(' Hz',6,1));

figure(2)
plot(ks_p,ftp,'Linewidth',[2],'Color','#019AFF'); hold on
plot(ks_r,ftr,'Linewidth',[2], 'Color','#E68A00' ); hold on
axis([150,1150,0,6000]) ;
text(note_freq,note_max,note_txt,'Color','k','FontSize',12);
legend('Piano','Recorder');
xlabel('Frequency (Hz)');
ylabel('|fft|');
title("Primary Frequencies of the Piano and Recorder");
```



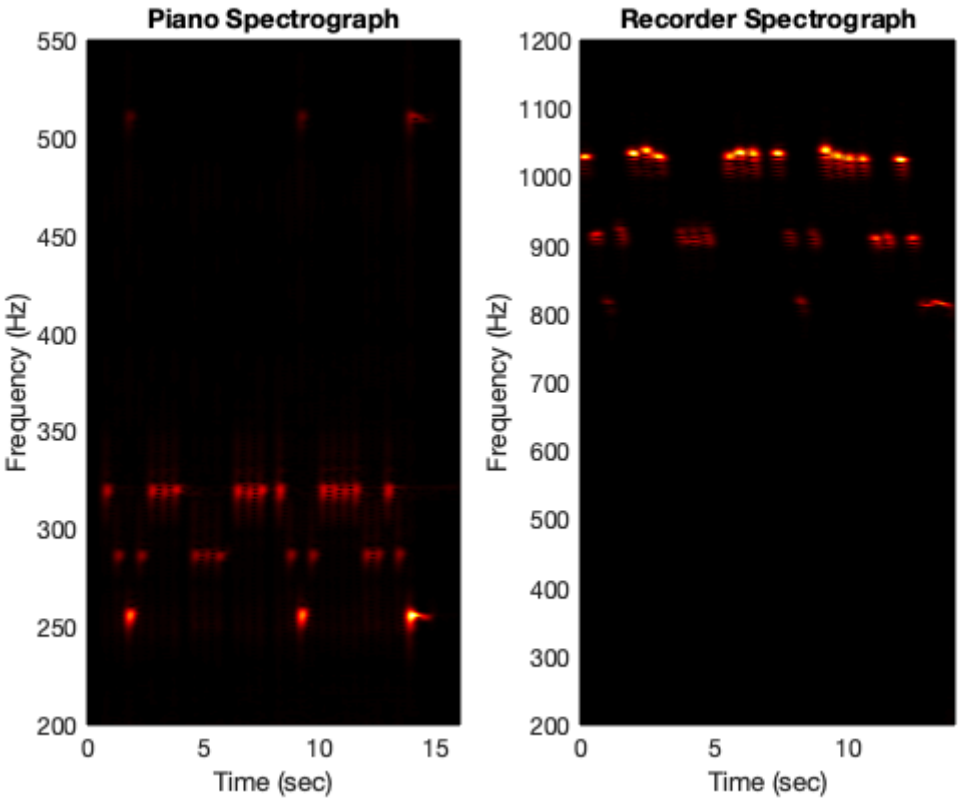
Spectrograph Data of Piano and Recorder

```
%Samples song at every .1 unit of time
```

```
width=20;
t_center_p=[0:.1:L_p];
t_center_r=[0:.1:L_r];
spec_p=[];
spec_r=[];

%piano spectrograph data
for i=1:length(t_center_p)
    g=exp(-width*(t_p-t_center_p(i)).^2);
    p_filt = g.*p;
    p_filt_transform = fft(p_filt);
    pt_plot = fftshift(abs(p_filt_transform));
    spec_p=[spec_p;pt_plot];
end
%recorder spectrograph data
for i=1:length(t_center_r)
    g=exp(-width*(t_r-t_center_r(i)).^2);
    r_filt = g.*r;
    r_filt_transform = fft(r_filt);
    rt_plot = fftshift(abs(r_filt_transform));
    spec_r=[spec_r;rt_plot];
end

%remove negative frequencies
spec_p(:,1:n_p/2-1)=[];
ks_p(1:n_p/2-1)=[];

spec_r(:,1:n_r/2-1)=[];
ks_r(1:n_r/2-1)=[];
```

Display Spectrographs

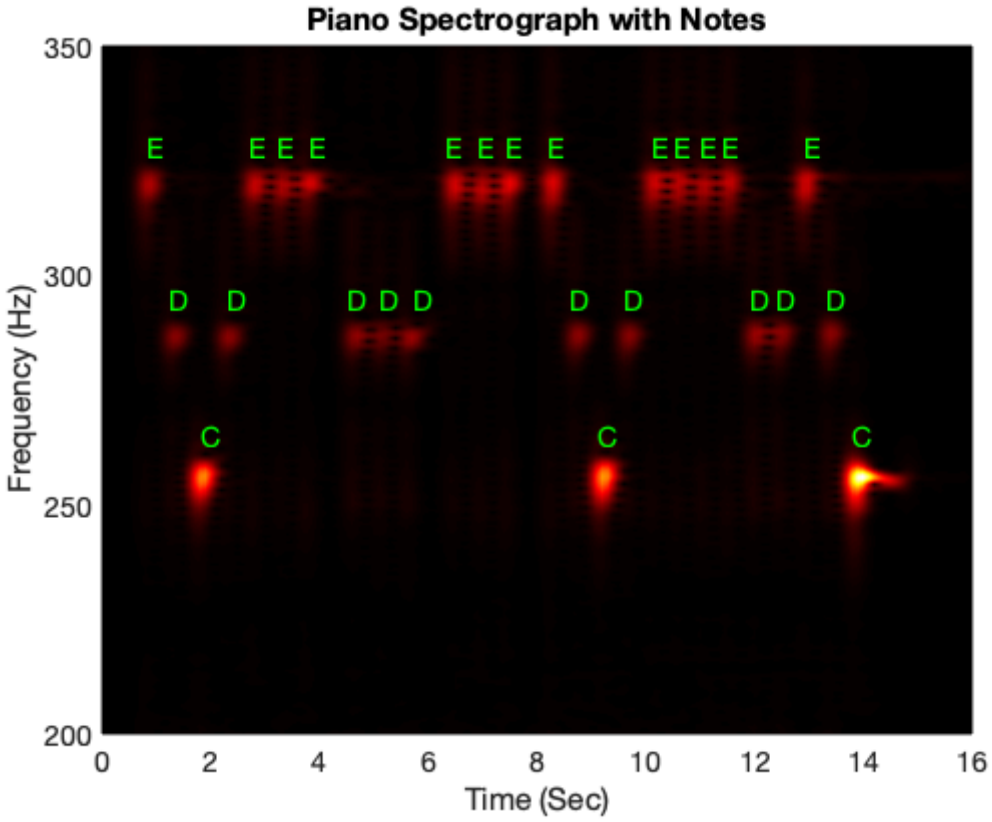
```
figure(3)
subplot(1,2,1)
pcolor(t_center_p,ks_p,spec_p.'),
shading interp ;
set(gca,'Ylim',[200 550],'Fontsize',[12]) ;
colormap(hot);
xlabel('Time (sec)'); ylabel('Frequency (Hz)');
title("Piano Spectrograph");

subplot(1,2,2)
pcolor(t_center_r,ks_r,spec_r.'),
shading interp ;
set(gca,'Ylim',[200 1200],'Fontsize',[12]) ;
colormap(hot);
xlabel('Time (sec)'); ylabel('Frequency (Hz)');
title("Recorder Spectrograph");
```

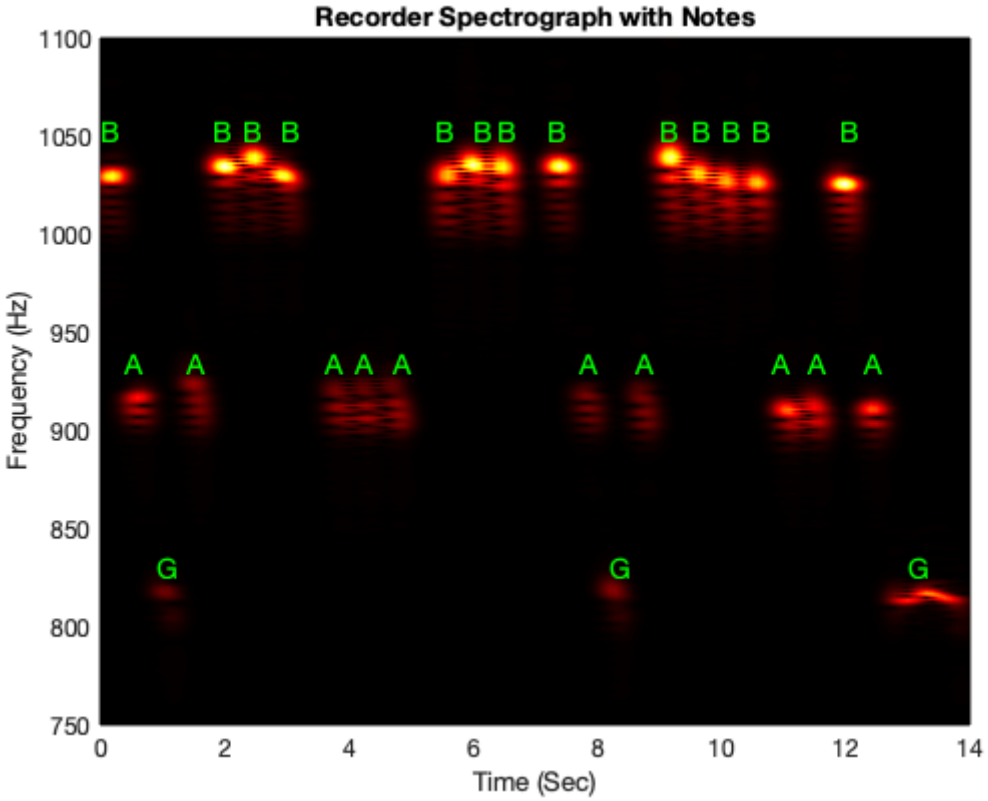


Display Spectrographs Zoomed in with Notes

```
px_e = [.8 2.7 3.2 3.8 6.3 6.9 7.4 8.2 10.1 10.5 11 11.4 12.9];
px_d = [1.2 2.3 4.5 5.1 5.7 8.6 9.6 11.9 12.4 13.3];
px_c = [1.8 9.1 13.8];

py_e = repelem(328,length(px_e));
py_d= repelem(295,length(px_d));
py_c= repelem(265,length(px_c));

figure(4)
%subplot(2,1,1)
pcolor(t_center_p,ks_p,spec_p.'),
shading interp
set(gca,'Ylim',[200 350],'Fontsize',[14])
colormap(hot)
xlabel('Time (Sec)'); ylabel('Frequency (Hz)');
title("Piano Spectrograph with Notes")
text(px_e,py_e,"E",'Color','g','FontSize',14)
text(px_d,py_d,"D",'Color','g','FontSize',14)
text(px_c,py_c,"C",'Color','g','FontSize',14)

% recorder
rx_e = [0 1.8 2.3 2.9 5.4 6 6.4 7.2 9 9.5 10 10.5 11.9];
rx_d = [.4 1.4 3.6 4.1 4.7 7.7 8.6 10.8 11.4 12.3];
```

```
rx_c = [.9 8.2 13];

ry_e = repelem(1053,length(rx_e));
ry_d= repelem(935,length(rx_d));
ry_c= repelem(831,length(rx_c));

figure(5)
%subplot(2,2,1)
pcolor(t_center_r,ks_r,spec_r.'),
shading interp
set(gca,'Ylim',[750 1100],'Fontsize',[12])
colormap(hot)
xlabel('Time (Sec)'); ylabel('Frequency (Hz)');
title("Recorder Spectrograph with Notes")
text(rx_e,ry_e,"B",'Color','g','FontSize',14)
text(rx_d,ry_d,"A",'Color','g','FontSize',14)
text(rx_c,ry_c,"G",'Color','g','FontSize',14)
```





Published with MATLAB® R2019b