Time-Frequency Analysis of Timeless Pieces

Lahari Gorantla
AMATH 482

Abstract

Music is a beautiful art form that changes its average Fourier components over a given time. Each time point tells a different story, which is why just a Fourier Transform over the whole piece is a limitation. To extract information of the piece, spectral methods can be utilized to decompose musical data in the time domain into its respective frequency components. This report uses the Gabor transform to perform a time-frequency analysis on pieces of music to create spectrograms and reconstruct music.

I. Introduction and Overview

At a given time point in a musical piece, it is difficult to understand the frequency profile of the music. Simply taking the Fourier Transform of the whole piece, disregards the change in average Fourier components over the time domain. The Gabor transformation breaks down the piece into small chunks in the time domain by applying a filter over the chunk and then performing a Fourier Transform on that chunk of time to decompose the piece into its respective frequency components. By creating a spectrogram of the musical data in the time domain and the frequency domain, one can see the visual representation of the musical data.

The first section of this report focuses on analyzing the visual representation of Handel's *Messiah* in the time-frequency domain by altering parameters like the sampling rate and the window of the filter. In addition, this report will be exploring the standard Gaussian filter, Mexican Hat wavelet, and the Shannon filter. The second section of this report focuses on music score reconstruction of *Mary Had a Little Lamb*, played on both a piano and a recorder. To reconstruct a clean score, there was spectral filtering around the fundamental frequencies to reduce the noise of the overtones.

II. Theoretical Background

Fourier Transform

Fourier transforms can transform can take a 2π-periodic signal over a time course or a signal over a spatial domain and decompose it into its constituent frequencies in the frequency domain, denoted by k.

$$F(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ikx} f(x) dx \qquad (1)$$

In MATLAB, the *fft* function computes the Discrete Fourier Transforms, which assumes discretized signal data rather than a continuous signal, using a fast Fourier transform algorithm.

This transform is ideal for characterizing a stationary signal that has an average signal value that does not change in time, and thus time information is eliminated in the transformation. For a nonstationary signal, extracting time and frequency information is important.

Gabor Transform

The Gabor transform, also known as short-time Fourier Transform, is a modification of the Fourier transform equation where the term g(τ-t) acts as a time filter to localize the signal over a specific window of time. The integration of this term slides the window down the signal to localize time chunks to decompose it into its respective frequencies. The term ω denotes the angular frequency of the signal, and the term τ slides the window.

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

Gabor Transform Filters

The standard Gabor transform filter uses the Gaussian filter in the time domain, where τ is the translation parameter and σ is the filter width parameter. This same filter can be utilized to as a low pass filter in the frequency domain that eliminates high-frequency components in the system of interest. However, the user needs to know what the desired frequency being investigated is, for the filter to work.

$$F(t) = e^{-\sigma(t-\tau)^2} \qquad (3)$$

Some alternative filters used for the Gabor transform include the Mexican Hat wavelet (equation 4) and the Shannon filter, which is a rectangular pulse function. They provide stronger attenuation of the undesired portion of the signal and stronger emphasis on the desired portion of the signal.

$$\psi(t) = (1 - t^2)e^{-t^2/2} \qquad (4)$$

Spectrograms

They are a visual representation of the time and frequency domains, where the color represents the intensity of the signal strength.

III. Algorithm Implementation and Development

Part 1:

Building the Function

After loading the music data file *handel* on Matlab, we find the spatial domain by taking the length of the music vector and dividing it by the sampling rate from the data file. The Fourier transform needs to be rescaled to the 2π domain and so the wavenumber, k, is multiplied by 2π/L and then shifted. As seen in *Equation 3*, a tslide vector centered around τ is created through a for loop to slide it through the continuous signal. As the shifting window centers around different τ values, the Gaussian filter with the tslide component is multiplied element-wise to the original signal to only capture the desired portion of the signal. That section of time is taken, Fourier transformed, taken the absolute value of to convert the complex components into real numbers,

and *fftshift*-ed to switch the two halves of the frequency domain. The for-loop creates a matrix with time and frequency data in it. It is then plotted as a spectrogram with the function *pcolor*. While plotting the spectrograms, the frequency domain value was divided by 2π to convert it into Hz units.

Tuning Parameters

To analyze Gabor transformation, the window width and sampling frequencies were varied. A window width of 1, 10, 100, and 100 were used for the Gaussian filter. In *Equation 3,* these values represent the σ and in the code, they are represented by a. To analyze under sampling, normal sampling, and oversampling, values of 1, 10, and 100 Hz were used, respectively. While testing the sampling values, a window length of 100 was chosen. While testing the window length, 10 Hz was chosen as the sampling frequency, in the code it is represented as time taken at 0.1 second increments.

Varying Filters

The Mexican Hat wavelet and Shannon filter were studied with varying window lengths, 0.01, 0.1, 1, and 10. The sampling frequency chosen was 10 Hz for all iterations. The Mexican Hat wavelet was built with *Equation 4* with modifications. To create tslide, the t is replaced by $(t-\tau)$ and this tslide is divided by the window length. The Shannon filter is created with two Heaviside functions to represent a rectangular pulse.

Part 2:

Removing Overtones

After loading in the music files containing *Mary Had a Little Lamb*, the Gabor Transform is built as described in Part 1, Building the Function. The standard Gaussian filter is used with a window of 100 and a sampling frequency of 10Hz. There are overtones present in the initial spectrogram. To get a clean spectrogram, a Gaussian filter in the frequency domain is used to find the center maximum frequency and attenuate the other frequencies. This center maximum frequency represents the fundamental frequency of the instrument.

IV. Computational Results

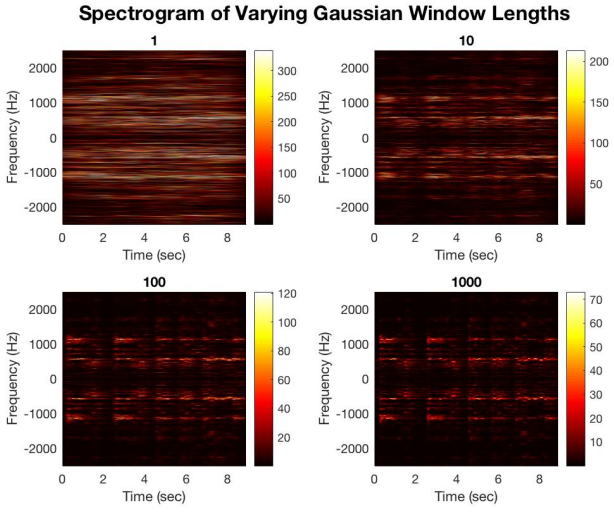

Figure 01. A compilation of spectrograms with varying Gaussian Window Lengths, 1, 10, 100, and 1000.

Increasing the window length of a Gaussian filter decreases the Gabor window at which the signal is localized in the time domain. As seen in Figure 01, this decrease in the Gabor window corresponds to an increase in the time resolution and a decrease in the frequency resolution.

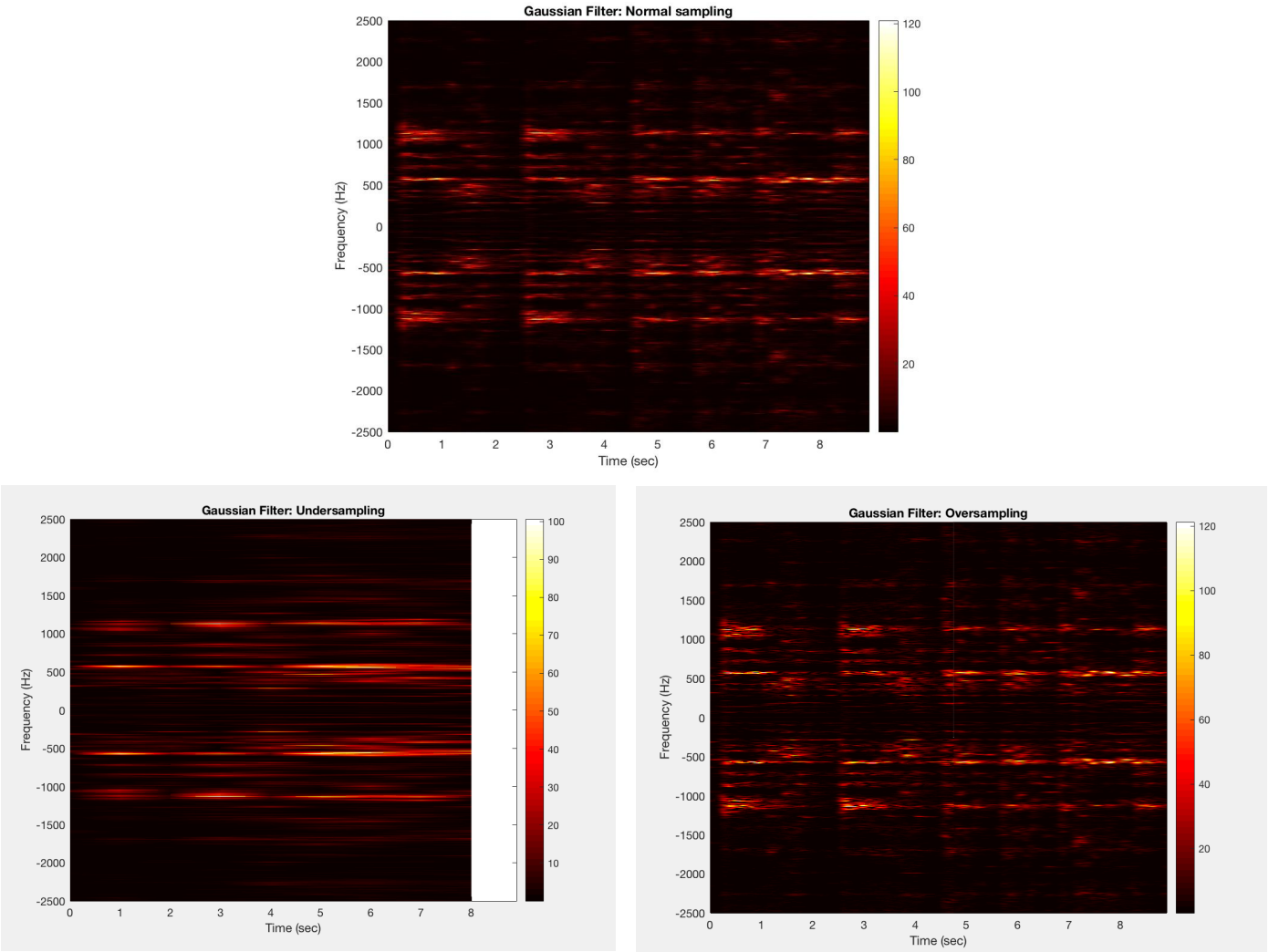

Figure 02. A compilation of spectrograms with varying sampling frequencies: normal sampling at 10Hz, under-sampling at 1Hz, and over-sampling at 100Hz.

As seen in Figure 02, oversampling provides a spectrogram with the most clarity. It has very distinct regions of signal strength. However, it is computationally inefficient. The normal sampling frequency used, 10 Hz, looked similar to that of the over-sampled spectrogram. The under-sampled spectrogram does not even gather information on the last .9 seconds of the clip. The resolution of the spectrogram is quite bad and difficult to interpret in both the time and frequency domain.

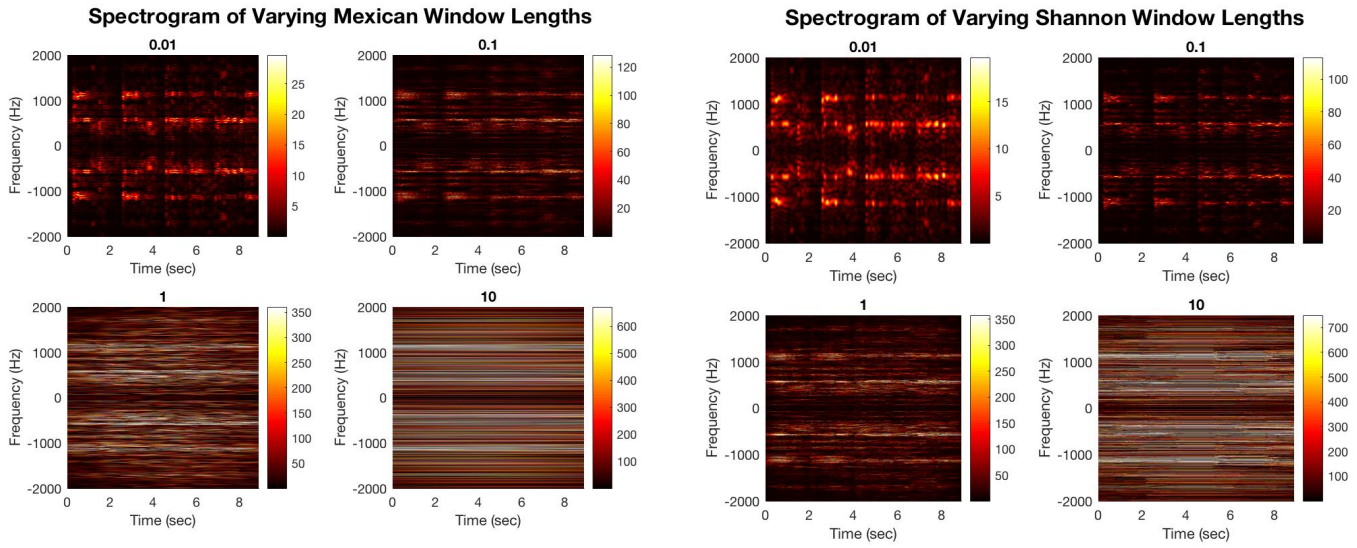

Figure 03. Spectrograms with the Mexican Hat Wavelet and Shannon filters with varying window lengths, 0.01, 0.1, 1, and 10. Each subplot is labeled with the respective window length used.

Increasing the window lengths for both the Mexican window length and Shannon window length increases the Gabor window used. As seen in Figure 03, the time resolution decreases while the frequency resolution increases. They look very similar. Although it appears that the Shannon filter has more accuracy in signal strength and in the frequency domain at the sacrifice of a little less time resolution.

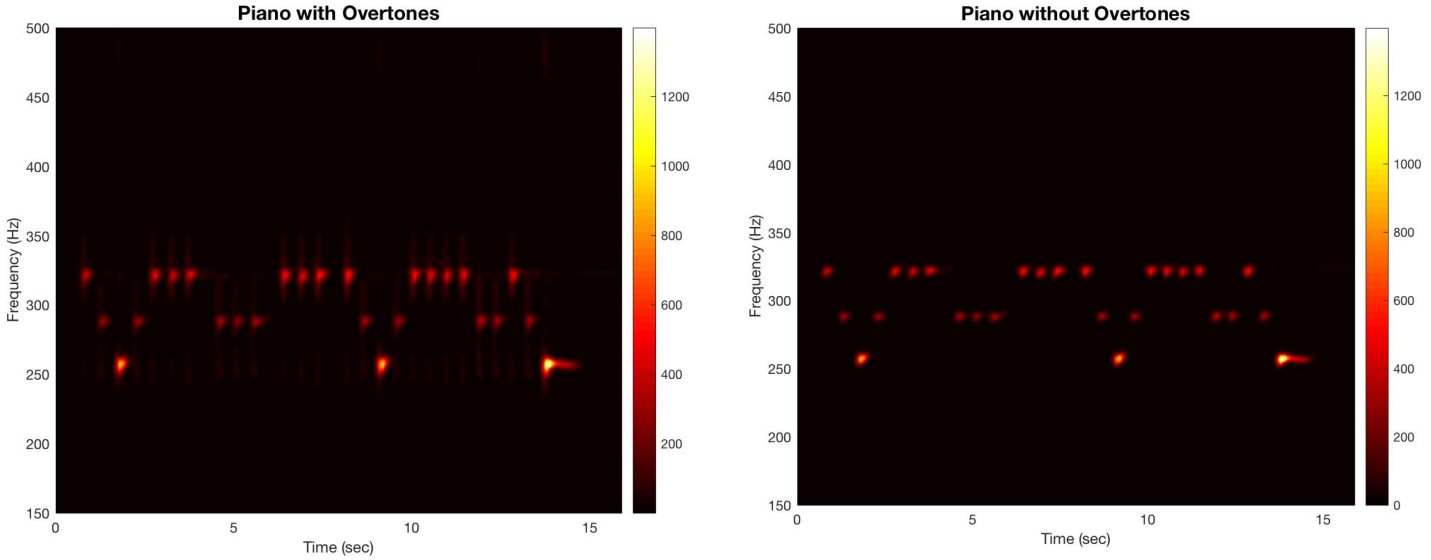

Figure 04. Recording of Mary Had a Little Lamb on the Piano with and without overtones.

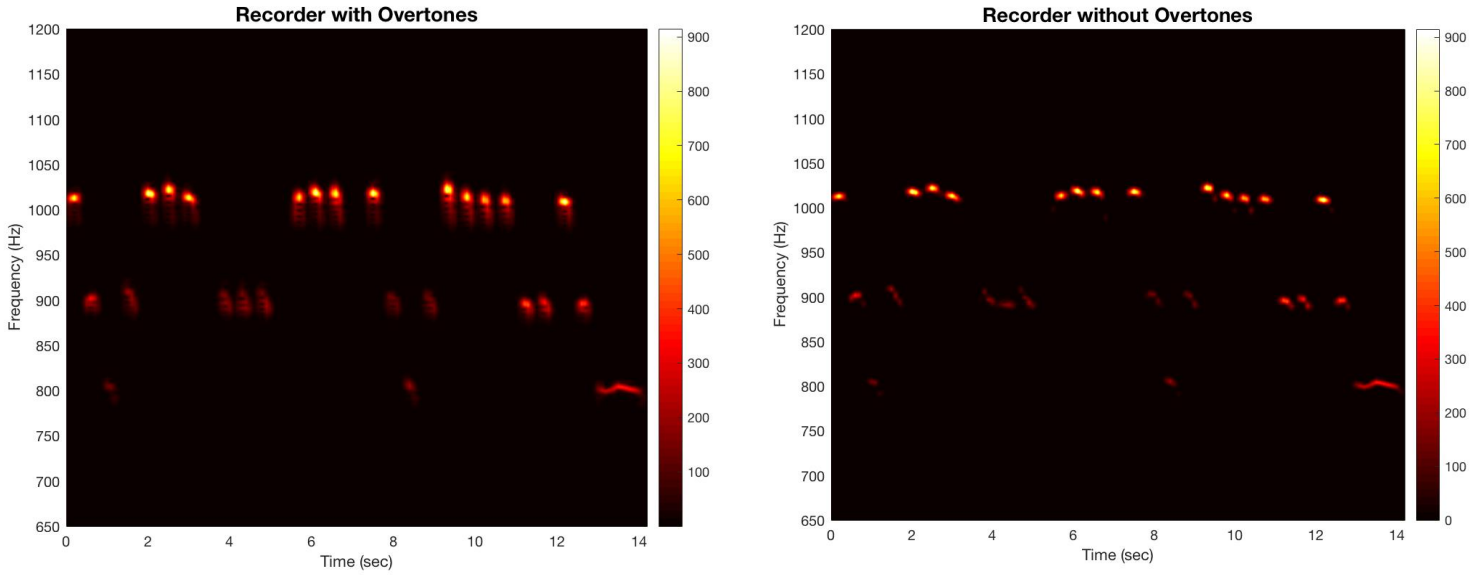

Figure 05. Recording of Mary Had a Little Lamb on the Piano with and without overtones.

As seen in Figures 04 and 05, there are little to no overtones in the musical pieces of *Mary Had a Little Lamb*. The "blur" effect around the center frequency is filtered out. It is more obvious to see that on the piano the first note starts at approximately 330Hz and the recorder starts at approximately 1000Hz.

V. Summary and Conclusion

The musical pieces, *Messiah*, and *Mary Had a Little Lamb*, were visually reconstructed in the frequency and time domain via the Gabor transform. As seen in the spectrograms for the Gaussian filter, Mexican Hat wavelet filter, and Shannon filter, there is a trade-off between frequency and time resolution. As the Gabor window increases, the time resolution decreases and the frequency resolution increases. Just like the Heisenberg principle, it is impossible to know both things at the same time. However, the Mexican Hat wavelet filter and Shannon filter were better compared to the Gaussian filter because of the nature of the filters. Shannon filter is a box that attenuates more precisely and the wavelet transform starts with a larger Fourier domain and then scales down. When it comes to sampling, under-sampling is obviously the worst. While there is a trade-off between computational efficiency and extreme precision between normal sampling and over-sampling.

Appendix A. MATLAB functions used and brief implementation explanation

abs(x): returns the absolute value of x; it was used to convert complex numbers into real numbers
fft(x): performs the Fast Fourier transform from time to frequency domain
fftshift(x): switches the left and right side of the frequency plot for accurate representation
pcolor(x): pseudocolor plot used to create a spectrogram

Appendix B. MATLAB Code

```
clear all; close all; clc;

%Loading the music
load handel
v = y';
figure(1); plot((1:length(v))/Fs,v);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Signal of Interest, v(n)');
p8 = audioplayer(v,Fs); playblocking(p8);
signal_trans = fftshift(fft(v));

%Setting up the domains
L=length(v)/Fs; n=length(v);
t2=linspace(0,L,n+1); t=t2(1:n);
k=(2*pi/L)*[0:(n+1)/2-1 -n/2:-1];
ks=fftshift(k);

figure(2) %The frequency plot
plot(ks,abs(signal_trans)/max(abs(signal_trans)));

%a = 100; defining the window width
count = 1;
tslide=0:.1:L; %list of tau values to sample
Sgt_spec = zeros(length(tslide),length(v));

for a = [1 10 100 1000] %Varying window length

    for j=1:length(tslide) %creating the spectrogram
        g=exp(-a*(t-tslide(j)).^2);
        Sg=g.*v;
        Sgt=fft(Sg);
        Sgt_spec(j,:) = fftshift(abs(Sgt));
    end

    %Making a plot
    subplot(2,2,count); hold on;
    sgtitle('Spectrogram of Varying Gaussian Window Lengths',...
        'FontSize',16,'FontWeight','bold');
    pcolor(tslide,ks/(2*pi),Sgt_spec.'),
    shading interp
    title(a, 'FontSize',14);
    ylabel('Frequency (Hz)', 'FontSize',12);
    xlabel('Time (sec)','FontSize',12);
    set(gca,'Ylim',[-2500 2500],'Xlim',[0 9.8],'Fontsize',10)
    colormap(hot)
    colorbar;
    count = count + 1;

end

clear all; close all; clc;

%Loading the music
load handel
v = y';
figure(1); plot((1:length(v))/Fs,v);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Signal of Interest, v(n)');
p8 = audioplayer(v,Fs); playblocking(p8);
signal_trans = fftshift(fft(v));

%Setting up the domains
L=length(v)/Fs; n=length(v);
t2=linspace(0,L,n+1); t=t2(1:n);
k=(2*pi/L)*[0:(n+1)/2-1 -n/2:-1];
ks=fftshift(k);

tslide=0:.1:L;
Sgt_spec = zeros(length(tslide),length(v));
a = 0.1;

%Mexican Hat Filter

for j=1:length(tslide) %creating the spectrogram
    mh_wavelet = (1-((t-tslide(j))/a).^2).*exp(-(((t-tslide(j))/a).^2) / 2);
```

```
    Sg=mh_wavelet.*v;
    Sgt=fft(Sg);
    Sgt_spec(j,:) = fftshift(abs(Sgt));
end

figure(2);
pcolor(tslide,ks/(2*pi),Sgt_spec.'),
shading interp
title('Mexican Hat Filter- window width: 0.1');
ylabel('Frequency (Hz)', 'FontSize',12);
xlabel('Time (sec)','FontSize',12);
%set(gca,'Ylim',[-500 500],'Xlim',[0 8.9],'Fontsize',10)
colormap(hot)
colorbar;


clear all; close all; clc;

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

p8 = audioplayer(v,Fs);
playblocking(p8);
signal_trans = fftshift(fft(v));

tslide=0:0.1:length(v)/Fs; %sampling
Sgt_spec = zeros(length(tslide),length(v));
window = 0.1; %window width
new_array = zeros(length(v),1);
L=length(v)/Fs;
n=length(v);
t2=linspace(0,L,n+1);
t=t2(1:n);
k=(2*pi/L)*[0:(n+1)/2-1 -n/2:-1];
ks=fftshift(k);
height = 1;
% The Shannon filter

for j=1:length(tslide)

    new_array = zeros(length(v),1);
    down = tslide(j)-(window/2);
    up = tslide(j)+(window/2);

    if down < 0 %creates a pulse
       B = t <= up;
       new_array(B') = height.*v(B');

    else
       B = t>=down & t<=up;
       new_array(B') = height.*v(B');

    end
    Sgt=fft(new_array);
    Sgt_spec(j,:) = fftshift(abs(Sgt));
end

figure(2);
pcolor(tslide,ks/(2*pi),Sgt_spec.'),
shading interp
title('Shannon Filter- window width: 0.1');
ylabel('Frequency (Hz)', 'FontSize',12);
xlabel('Time (sec)','FontSize',12);
%set(gca,'Ylim',[-2500 2500],'Xlim',[0 8.9],'Fontsize',10)
colormap(hot)
colorbar;

clear all; close all; clc;

%Loading the music
load handel
v = y';
figure(1); plot((1:length(v))/Fs,v);
xlabel('Time [sec]'); ylabel('Amplitude');
```

```
title('Signal of Interest, v(n)');
p8 = audioplayer(v,Fs); playblocking(p8);
signal_trans = fftshift(fft(v));

%Setting up the domains
L=length(v)/Fs; n=length(v);
t2=linspace(0,L,n+1); t=t2(1:n);
k=(2*pi/L)*[0:(n+1)/2-1 -n/2:-1];
ks=fftshift(k);

tslide=0:.1:L;
Sgt_spec = zeros(length(tslide),length(v));
%a = 0.1;
count = 1;

%Mexican Hat Filter

for a = [0.01 0.1 1 10] %Varying window length


    for j=1:length(tslide) %creating the spectrogram
        mh_wavelet = (1-((t-tslide(j))/a).^2).*exp(-(((t-tslide(j))/a).^2) / 2);
        Sg=mh_wavelet.*v;
        Sgt=fft(Sg);
        Sgt_spec(j,:) = fftshift(abs(Sgt));
    end

    %Making a plot
    subplot(2,2,count); hold on;
    sgtitle('Spectrogram of Varying Mexican Window Lengths',...
        'FontSize',16,'FontWeight','bold');
    pcolor(tslide,ks/(2*pi),Sgt_spec.'),
    shading interp
    title(a, 'FontSize',14);
    ylabel('Frequency (Hz)', 'FontSize',12);
    xlabel('Time (sec)','FontSize',12);
    set(gca,'Ylim',[-2000 2000],'Xlim',[0 8.9],'Fontsize',10)
    colormap(hot)
    colorbar;
    count = count + 1;

end

clear all; close all; clc;

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

p8 = audioplayer(v,Fs);
playblocking(p8);
signal_trans = fftshift(fft(v));

tslide=0:0.1:length(v)/Fs; %sampling
Sgt_spec = zeros(length(tslide),length(v));
%indow = 0.1; %window width
new_array = zeros(length(v),1);
L=length(v)/Fs;
n=length(v);
t2=linspace(0,L,n+1);
t=t2(1:n);
k=(2*pi/L)*[0:(n+1)/2-1 -n/2:-1];
ks=fftshift(k);
height = 1;
count =1;
%Shannon Filter

for a = [0.01 0.1 1 10] %Varying window length

    for j=1:length(tslide)

        new_array = zeros(length(v),1);
        down = tslide(j)-(a/2);
        up = tslide(j)+(a/2);

        if down < 0 %creates a pulse
```

```
            B = t <= up;
            new_array(B') = height.*v(B');

        else
            B = t>=down & t<=up;
            new_array(B') = height.*v(B');

        end
        Sgt=fft(new_array);
        Sgt_spec(j,:) = fftshift(abs(Sgt));
    end

    %Making a plot
    subplot(2,2,count); hold on;
    sgtitle('Spectrogram of Varying Shannon Window Lengths',...
        'FontSize',16,'FontWeight','bold');
    pcolor(tslide,ks/(2*pi),Sgt_spec.'),
    shading interp
    title(a, 'FontSize',14);
    ylabel('Frequency (Hz)', 'FontSize',12);
    xlabel('Time (sec)','FontSize',12);
    set(gca,'Ylim',[-2000 2000],'Xlim',[0 8.9],'Fontsize',10)
    colormap(hot)
    colorbar;
    count = count + 1;

end

clear all; close all; clc;

%Loading the music
load handel
v = y';
figure(1); plot((1:length(v))/Fs,v);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Signal of Interest, v(n)');
p8 = audioplayer(v,Fs); playblocking(p8);
signal_trans = fftshift(fft(v));

%Setting up the domains
L=length(v)/Fs; n=length(v);
t2=linspace(0,L,n+1); t=t2(1:n);
k=(2*pi/L)*[0:(n+1)/2-1 -n/2:-1];
ks=fftshift(k);

figure(2) %The frequency plot
plot(ks,abs(signal_trans)/max(abs(signal_trans)));

%a = 100; defining the window width
count = 1;
tslide=0:.01:L; %list of tau values to sample
Sgt_spec = zeros(length(tslide),length(v));
a = 100;

for j=1:length(tslide) %creating the spectrogram
        g=exp(-a*(t-tslide(j)).^2);
        Sg=g.*v;
        Sgt=fft(Sg);
        Sgt_spec(j,:) = fftshift(abs(Sgt));
end

pcolor(tslide,ks/(2*pi),Sgt_spec.'),
shading interp
title('Gaussian Filter: Undersampling', 'FontSize',14);
ylabel('Frequency (Hz)', 'FontSize',12);
xlabel('Time (sec)','FontSize',12);
set(gca,'Ylim',[-2500 2500],'Xlim',[0 8.9],'Fontsize',10)
colormap(hot)
colorbar;
count = count + 1;


clear all; close all; clc;

[y,Fs] = audioread('music1.wav');
tr_piano=length(y)/Fs; % record time in seconds
%plot((1:length(y))/Fs,y);
%xlabel('Time [sec]'); ylabel('Amplitude');
```

```
%title('Mary had a little lamb (piano)');
p8 = audioplayer(y,Fs); playblocking(p8);
%piano_freq = fftshift(fft(y));

%setting up the domain
L=length(y)/Fs; n=length(y);
t2=linspace(0,L,n+1); t=t2(1:n);
k=(2*pi/L)*[0:n/2-1 -n/2:-1];
ks=fftshift(k);

y=y';

%creating the slide
tslide=0:.1:length(y)/Fs;
Sgt_spec = zeros(length(tslide),length(y));
a = 100;

for j=1:length(tslide)
    %guassian window
    g=exp(-a*(t-tslide(j)).^2);
    Sg=g.*y;
    Sgt=fft(Sg);
    %filtering around fundamental frequency
    %[center_strength,index] = max(abs(Sgt));
    %center_freq = k(index);
    %filter = exp(-0.001*((k - center_freq).^2));
    %Sgt=filter.*Sgt;

    Sgt_spec(j,:) = fftshift(abs(Sgt));
end

figure(1);
pcolor(tslide,ks/(2*pi),Sgt_spec.')
set(gca,'Ylim',[150 500],'Fontsize',10)
title('Piano with Overtones','FontSize',14)
xlabel('Time (sec)');
ylabel('Frequency (Hz)');
shading interp
colormap(hot)
colorbar;


clear all; close all; clc;

[y,Fs] = audioread('music2.wav');
tr_rec=length(y)/Fs; % record time in seconds
%plot((1:length(y))/Fs,y);
%xlabel('Time [sec]'); ylabel('Amplitude');
%title('Mary had a little lamb (recorder)');
p8 = audioplayer(y,Fs); playblocking(p8);
%rec_freq = fftshift(fft(y));

%setting up the domain
L=length(y)/Fs; n=length(y);
t2=linspace(0,L,n+1); t=t2(1:n);
k=(2*pi/L)*[0:n/2-1 -n/2:-1];
ks=fftshift(k);

y=y';

%creating the slide
tslide=0:.1:length(y)/Fs;
Sgt_spec = zeros(length(tslide),length(y));
a = 100;

for j=1:length(tslide)
    %guassian window
    g=exp(-a*(t-tslide(j)).^2);
    Sg=g.*y;
    Sgt=fft(Sg);
    %filtering around fundamental frequency
    [center_strength,index] = max(abs(Sgt));
    center_freq = k(index);
    filter = exp(-0.001*((k - center_freq).^2));
    Sgt=filter.*Sgt;

    Sgt_spec(j,:) = fftshift(abs(Sgt));
end
```

```
figure(1);
pcolor(tslide,ks/(2*pi),Sgt_spec.')
set(gca,'Ylim',[650 1200],'Fontsize',10)
title('Recorder without Overtones','FontSize',14)
xlabel('Time (sec)');
ylabel('Frequency (Hz)');
shading interp
colormap(hot)
colorbar;
```