Time-Frequency Analysis on Music Clips

Makenna Barton - bartonmj@uw.edu

February 2020

Abstract This report outlines the theories and algorithms used to filter music signals and identify the notes being played. It also explores the ideas of over-sampling and under-sampling signals with filters, the Fourier Transform and the Gabor Transform. The choice of filter (using a Morelt wavelet filter, through the continuous wavelet transform, as compared to a Gaussian), filter width and filter shift and how those characteristics impact spectrograms is investigated though analysis on a Handel score.

1 Introduction and Overview

There are two separate applications of Time-Frequency Analysis within this report. Part 1 being exploration on a portion of a Handel score, looking at spectrograms after Gabor filtering. More closely, working to understand how changing the width of the filtering function effects the spectrogram, the choice of window size/sliding, what the outcome of oversampling versus under-sampling looks like in a spectrogram, and how the results are impacted by the choice of filter function. Part 2 reads in two pieces of music, one being Mary had a little lamb on piano and the other the same song played on the recorder. Then, through Gabor filtering, we will work to identify the notes played and when they were played in relation to each other to reproduce the score. Finally, using Time-Frequency analysis we will look to find the differences between the piano and recorder based on their frequencies and the unique timbre of each instrument.

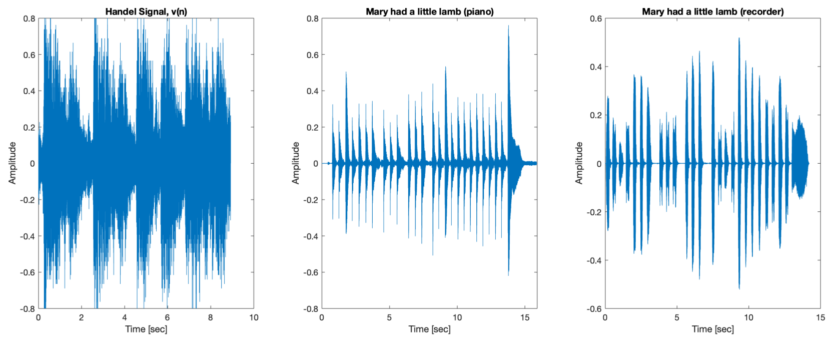

Figure 1: Original, unfiltered, music signal data

2 Theoretical Background

The Fourier Transform is a powerful tool to understand signals in the frequency domain. A drawback of this Transform is that you lose all information on the time domain. When looking at non-stationary signals (frequencies non-constant in time), such as music signals, we want to know information about both time and frequency. A strategy to accomplish this is to look at the signal through a filter on a specific time window, and then shift that window along the length of the signal. This strategy describes the Gabor Transform. The Gabor Transform is also known as the Short-Time Fourier Transform due to how similar their formulas are. Recall the Fourier Transform:

$$\hat{f}(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt \tag{1}$$

The formula for the Gabor Transform is as follows:

$$\tilde{f}(\tau, \omega) = \int_{-\infty}^{\infty} f(t) g(t-\tau) e^{-i\omega t} dt \tag{2}$$

where τ is the center of the filter and $g(t-\tau)$ is the filter function. It is common to use a g filter function that is normalized, real, and symmetric such as a Gaussian (which we used to solve the problems outlined in this report - see Equation 4). The Inverse Gabor Transform is:

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

The Gabor Transform and its inverse are used to take a filter in time, transform to get information about the frequency domain over that time interval, and transform back to the original domain. This allows us to understand how the frequency is changing over time. While this method is incredibly useful and can be customized by the choice of filter function, we must consider its limitations. How the "width" of the filter window (a), paired with how we slide along the function in time ($\Delta\tau$) can impact the information we get back from the Transform. When the window the filter function covers is very wide, you get back information solely on frequency, and you have no time information. This is the Fourier Transform. When the filter function is focused on a very thin window, you have very detailed information about the time at that point on the function but very little about the frequency. A variation between very thin and very wide is called Time-Frequency Analysis. Using Gabor with an appropriately defined a and $\Delta\tau$ can give us adequate information on both time and frequency.

$$F(k) = exp(-a(t-\tau)^2) \tag{4}$$

An example of a Gaussian filter sliding across a signal is shown in Figure 2.

The concepts of oversampling and undersampling are import considerationd to make when conducting Time-Frequency Analysis. Oversampling occurs when

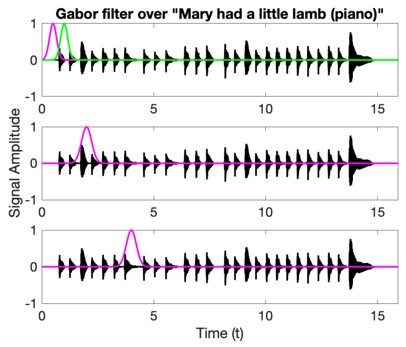

Figure 2: Gaussian filter shifting over time as steps of the Gabor Transform

the time steps between filters are very small in conjunction with a wider filter. In mathematical terms, when the filter function has a small (a), the width of the filter increases, and when $\Delta\tau$ is small, the filters have a large amount of overlap, causing oversampling of the function. Undersampling is the inverse. When the filter is narrow and/or the time steps are too large, also known as large (a) and small $\Delta\tau$, the filter could miss portions of the signal, therefore undersampling the data.

3 Algorithm Implementation and Development

While exploring a famous clip of music from Handel, I created spectrograms in MATLAB to compare how filter size and shape would impact the time-frequency signature of the piece. I figured the best way to compare width size (a) to filter shift $(\Delta\tau)$ was creating a vector of four possible (a) values, $[20, 10, 5, 1]$. I then selected a set of $\Delta\tau$ values, $[.01, .1, 1]$, and for each value of $\Delta\tau$ I computed the Gabor transform at each value of (a) and plotted spectrograms. This process can be seen in Figures 3. These ranges of $\Delta\tau$ and (a) and their result as a filter function over a signal allowed me to see a spectrum of oversampling to undersampling. Finally, to see how different filters affect the spectrogram I used the Continuous Wavelet Transform on the music signal and compared it to the results of the Gaussian filter.

To approach the problem of identifying the notes being played within the two renditions of Mary had a little lamb, I first considered the signals in a time versus amplitude graph (see Figure 1). Doing so allowed me to get a general sense for the songs, visually, through aspects like how many notes were played and how long they were held. Next, I implemented a Gabor transform, using a Gaussian filter, along the length of time for the clip. Through guess and check, I determined that looking at the functions in Time versus Frequency space gave the most clear information on both variables when the a (width) of the filter was 10 and the $\Delta\tau$ was 0.1. After looking at the Time-Frequency Spectrogram for each song (Figure 6), I was able to determine which notes were being played by converting their frequencies in Hertz to musical notes.

4 Computational Results

Through my investigation of Handel's music, I was able to draw conclusions about how oversampling and undersampling are related to the choice of filter width (a) and how far the filter is translated at each time step $(\Delta\tau)$. From the spectrograms created (Figure 3) I was able to deduce that the choice of a and $\Delta\tau$ are inversely related. Moreover, when a is increasing, the width

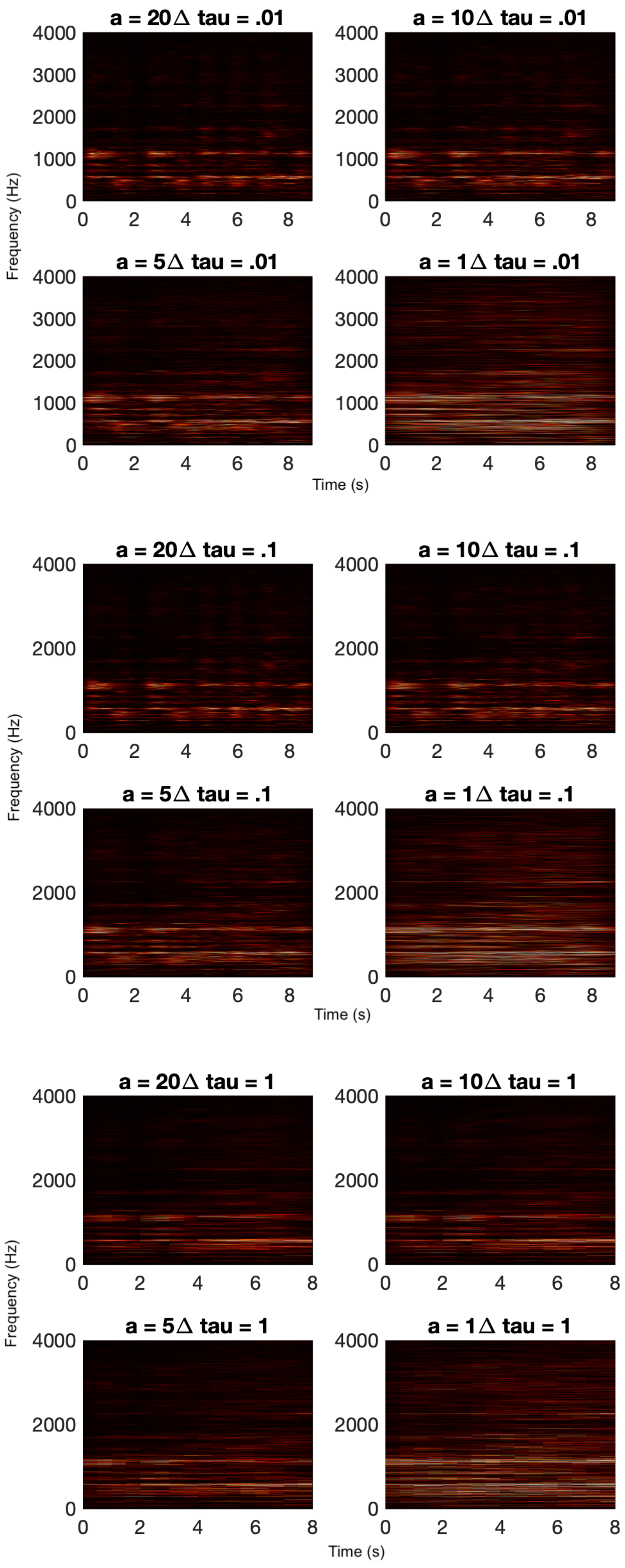

Figure 3: Spectrograms of the Handel clip. Varying a values for the given $\Delta\tau$ in each set of four figures. They show how a and $\Delta\tau$ are related to each other

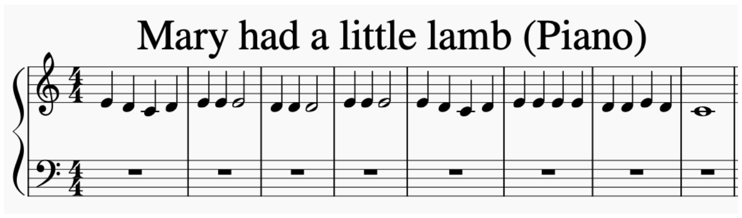

Figure 4: Reproduction of Mary had a little lamb (Piano, music1.wav)

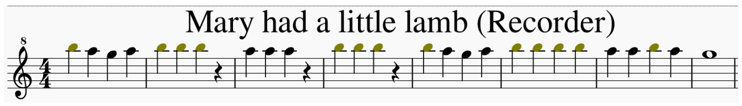

Figure 5: Reproduction of Mary had a little lamb (Recorder, music2.wav)

of the filter is decreasing, therefore the distance between the steps ($\Delta\tau$) must decrease to accommodate the skinnier filter and capture all the data. The inverse is true as well, if $\Delta\tau$ is increasing, the distance between the filter steps is increasing, therefore the filter must widen to ensure sampling of all the signal data. Oversampling occurs when both a and $\Delta\tau$ are small, see Figure 3 with $a = 1$ and $\Delta\tau = .01$. The horizontal lines indicate there is detailed information in the frequency domain because the data is being transformed so many times, but precision in the time domain is lost from oversampling. Undersampling occurs when both a and $\Delta\tau$ are big, see Figure 3 with $a = 20$ and $\Delta\tau = 1$. The figure lacks detail because much of the data is being missed between each time step. I compared the built-in Continuous Wavelet Transform function in MATLAB to see what a different form of filtering on the same signal would look like (see Figure 7). The shape of the Scalogram is caused from the way wavelets are applied to signals, a process called Multi-Resolution Analysis. The detail of time and frequency at the beginning is very low, but then it allows both time and frequency to be very precise at later iterations. It is clear to me that even the use of the Gabor Transform in Figure 3 with most detail doesn't compare to the precision the CWT provides in time and frequency. The Gabor Transform with a Gaussian filter helps to denoise a signal to some extent, but I found other forms of filters can fit this data better.

After filtering the piano score of Mary had a little lamb over time I was able to determine what notes were playing and when. The song consisted of a variation of three notes: C_4 ($\approx$ 261.63 Hz - Middle C), D_4 ($\approx$ 293.66 Hz), E_4

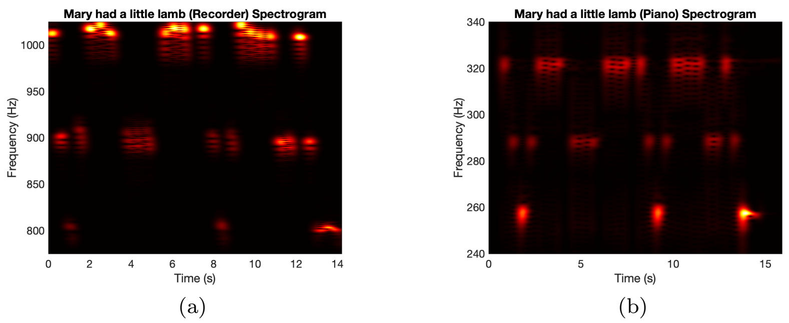

Figure 6: Spectrograms of Recorder and Piano renditions of Mary had a little lamb.

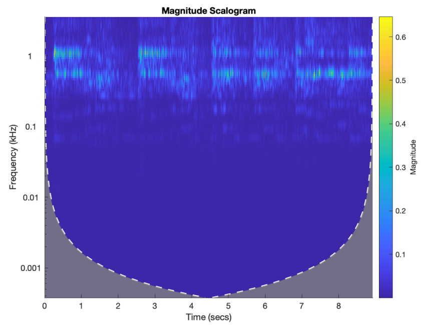

Figure 7: Continuous Wavelet Transform of Handel

($\approx$ 329.63 Hz). See Figure 4 for the whole score. Through Gabor Filtering on the recorder rendition of Mary had a little lamb, I determined the notes being played were: G_5 ($\approx$ 783.99 Hz), A_5 ($\approx$ 880.00 Hz), B_5 ($\approx$ 987.77 Hz). Although the notes being played are quite different there is one similarity that makes the songs sound the same. In both sets of three notes, the difference in Hertz between the frequencies is relatively constant. Meaning, the three notes on the piano are approximately 30 Hz apart from one another and on the recorder, the notes are about 100 Hz apart from the adjacent notes. As shown in Figure 6, the recorder has significantly more overtones in its timbre compared to the piano. This is shown in the way there is stronger definition of the overtones in the frequency domain, it appears as though many frequencies are being played at a single time. Those frequencies are unique to the recorder.

5 Summary and Conclusions

Exploration of Handel's music showed the importance of carefully selecting the parameters of the filter function used, as it can lead to either oversampling or undersampling the data leading to skewed results. Additionally, we saw the power of the Gabor transform and its ability to provide detail in both the time and frequency domains. But when compared to the Continuous Wavelt Transform for the Handel music, the Gaussian Filter in the Gabor Transform did not compare in precision. Using a Gaussian Filter proved valuable when used to single out frequencies from a piece of music. We were able to recreate two versions of Mary had a little lamb through identifying frequencies and converting them to notes.

6 Appendix A - MATLAB functions used and brief implementation explanation

`fft()`- Fast Fourier Transform - This function performs the Fourier Transform on the function you specify.

`load()`- This function loads in preexisting MATLAB data, the Handel music in our case.

`audioread()`- Reads in audio files and puts the data in a vector.

`cwt()`- Continuous Wavelet Transform - computes the CWT of the function you specify. Defaults to the Morse wavelet with the symmetry parameter (gamma) equal to 3 and the time-bandwidth product equal to 60.

`pcolor()`- Plotting function to create Spectrograms.

7 Appendix B - MATLAB Codes

Matlab Code

```
%% Part 1
clear; clc; close all;

load handel
v = y';

% Plot of all the unfiltered music signals
subplot(1,3,1)
plot((1:length(v))/Fs,v);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Handel Signal, v(n)');

subplot(1,3,2)
[S,Fs] = audioread('music1.wav');
S = S';
tr_piano=length(S)/Fs;  % record time in seconds
plot((1:length(S))/Fs,S);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (piano)');

subplot(1,3,3)
[S,Fs] = audioread('music2.wav');
tr_rec=length(S)/Fs;  % record time in seconds
plot((1:length(S))/Fs,S);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (recorder)');

%% Explore Handel
clear; clc; close all;
load handel
v = y';
S = v;
tr_handel=length(S)/Fs;
p8 = audioplayer(v,Fs);
playblocking(p8);

n = length(S);
L = tr_handel;
t = (1:length(S))/Fs ;
k = (1/L)*[0:(n-1)/2 -(n-1)/2:-1];
```

```

cwt(v,Fs)
print(gcf,'-dpng','cwt.png')

%% Spectrograms for varying a
a_vec = [20 10 5 1];
for jj = 1:length(a_vec)
    a = a_vec(jj);
    tslide=0:0.01:L;
    Sgt_spec = zeros(length(tslide),n);
    for j=1:length(tslide)
        g=exp(-a*(t-tslide(j)).^2);
        Sg=g.*S;
        Sgt=fft(Sg);
        Sgt_spec(j,:) = fftshift(abs(Sgt));
    end
    subplot(2,2,jj)
    pcolor(tslide,fftshift(k),Sgt_spec.'),
    shading interp
    title(['a = ',num2str(a),'\Delta tau = .01'],'Fontsize',16)
    set(gca,'Ylim',[0 4000],'Fontsize',16)
    colormap(hot)
end

figure(2)
a_vec = [20 10 5 1];
for jj = 1:length(a_vec)
    a = a_vec(jj);
    tslide=0:.1:L;
    Sgt_spec = zeros(length(tslide),n);
    for j=1:length(tslide)
        g=exp(-a*(t-tslide(j)).^2);
        Sg=g.*S;
        Sgt=fft(Sg);
        Sgt_spec(j,:) = fftshift(abs(Sgt));
    end
    subplot(2,2,jj)
    pcolor(tslide,fftshift(k),Sgt_spec.'),
    shading interp
    title(['a = ',num2str(a),'\Delta tau = .1'],'Fontsize',16)
    set(gca,'Ylim',[0 4000],'Fontsize',16)
    colormap(hot)
end
print(gcf,'-dpng','spectrogram_tau_1.png')

figure(3)
```

```
a_vec = [20 10 5 1];
for jj = 1:length(a_vec)
    a = a_vec(jj);
    tslide=0:1:L;
    Sgt_spec = zeros(length(tslide),n);
    for j=1:length(tslide)
        g=exp(-a*(t-tslide(j)).^2);
        Sg=g.*S;
        Sgt=fft(Sg);
        Sgt_spec(j,:) = fftshift(abs(Sgt));
    end
    subplot(2,2,jj)
    pcolor(tslide,fftshift(k),Sgt_spec.'),
    shading interp
    title(['a = ',num2str(a),'\Delta tau = 1'],'Fontsize',16)
    set(gca,'Ylim',[0 4000],'Fontsize',16)
    colormap(hot)
end
print(gcf,'-dpng','spectrogram_tau1.png')

%% Part 2
clear; clc; close all;
[S,Fs] = audioread('music1.wav');
S = S';
tr_piano=length(S)/Fs;  % record time in seconds
plot((1:length(S))/Fs,S);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (piano)');
p8 = audioplayer(S,Fs); playblocking(p8);

St = fft(S);
n = length(S);
L = tr_piano;
t = (1:length(S))/Fs ;
k = (1/L)*[0:(n)/2-1 -n/2:-1];

% Plot of what the gabor filter looks like on the signal over time
tau = 1;
a = 10;
g = exp(-a*(t-tau).^2);
tau = 1.5;
a = 10;
g1 = exp(-a*(t-tau).^2);
tau = 2;
a = 10;
g2 = exp(-a*(t-tau).^2);
```

```
figure(2)
subplot(3,1,1)
plot(t,S,'k',t,g,'m',t,g1,'g',t,g2,'c','Linewidth',2)
set(gca,'Fontsize',16), title('Gabor filter over "Mary had a little lamb (piano)
tau = 4;
a = 10;
g = exp(-a*(t-tau).^2);

subplot(3,1,2)
plot(t,S,'k',t,g,'m','Linewidth',2)
set(gca,'Fontsize',16), ylabel('Signal Amplitude')
tau = 6;
a = 10;
g = exp(-a*(t-tau).^2);

subplot(3,1,3)
plot(t,S,'k',t,g,'m','Linewidth',2)
set(gca,'Fontsize',16), xlabel('Time (t)')
print(gcf,'-dpng','gabor_filter.png')

% Calculate Gabor transform and plot spectrogram
a = 10;
tslide = 0:.1:L;
Sgt_spec = zeros(length(tslide),n);
frequencies = zeros(length(tslide),1);
for j = 1:length(tslide)
    g = exp(-a*(t-tslide(j)).^2);
    Sg = g.*S;
    Sgt = fft(Sg);
    Sgt_spec(j,:) = fftshift(abs(Sgt)); % We don't want to scale it
    frequencies(j) = min(abs(Sgt_spec(j,:)));
end

figure(3)
pcolor(tslide,fftshift(k),Sgt_spec.'),
shading interp
set(gca,'Ylim',[240 340],'Fontsize',16)
colormap(hot)
title('Mary had a little lamb (Piano) Spectrogram'),
xlabel('Time (s)'), ylabel('Frequency (Hz)')
print(gcf,'-dpng','piano_spectrogram.png')

% Recorder Analysis
clear; clc; close all;
figure(4)
[S,Fs] = audioread('music2.wav');
```

```
S = S';
tr_rec=length(S)/Fs;  % record time in seconds
plot((1:length(S))/Fs,S);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (recorder)');
p8 = audioplayer(S,Fs); playblocking(p8);

n = length(S);
L = tr_rec;
t = (1:length(S))/Fs ;
k = (1/L)*[0:(n)/2-1 -n/2:-1];

% Calculate Gabor transform and plot spectrogram
a = 10;
tslide = 0:.1:L;
Sgt_spec = zeros(length(tslide),n);
frequencies = zeros(length(tslide),1);
for j = 1:length(tslide)
    g = exp(-a*(t-tslide(j)).^2);
    Sg = g.*S;
    Sgt = fft(Sg);
    Sgt_spec(j,:) = fftshift(abs(Sgt)); % We don't want to scale it
    frequencies(j) = min(abs(Sgt_spec(j,:)));
end

figure(5)
pcolor(tslide,fftshift(k),Sgt_spec.'),
shading interp
set(gca,'Ylim',[775 1025],'Fontsize',16)
colormap(hot)
title('Mary had a little lamb (Recorder) Spectrogram'),
xlabel('Time (s)'), ylabel('Frequency (Hz)')
print(gcf,'-dpng','recorder_spectrogram.png')
```