Gábor Transforms

Jason Miller
AMATH 482

February 7, 2020

Abstract

* In this assignment we analyze audio recordings of music with time-frequency analysis. First, we explore the time-frequency signature of a clip of Handel's Messiah by making spectrograms with Gaussian filters, Mexican Hat wavelets, and step-functions. Next, we reproduce the music score of 'Mary Had a Little Lamb' played on both the piano and recorder using Gábor filtering, and use this to see the timbral difference between a recorder and a piano.

1 Introduction and Overview

1.1 Part One

We are given a portion of Handel's Messiah, and our goal is to perform time-frequency analysis on it. Time-frequency analysis is a body of techniques to analyze signals in both its time and frequency domains. This can provide valuable information about audio signals such as the ones we are given because they express different frequencies at different points in time. In order to complete this task, we first create spectrograms of the signal, and examine the affects of varying the window width and window translation size. Next, we do similar analysis using different Gábor windows, namely Mexican Hat Wavelets and Shannon windows.

1.2 Part Two

Now we are given two clips of 'Mary Had a Little Lamb', one played on piano, and the other played on the recorder. We want to reproduce the score of the music for each audio signal and determine the difference between pianos and recorders in terms of the overtones that

they produce when played. We do this by creating spectrograms using Gaussian windows and filtering out the overtones to clean up the score.

2 Theoretical Background

Recall that the Fourier Transform is an operator that decompose a signal into its frequency components, and is defined as:

$$\hat{f}(k) = \int_{\mathbb{R}} f(x)e^{-ikx}dx \tag{1}$$

$$f(x) = \int_{\mathbb{R}} \hat{f}(k)e^{ikx}dk \tag{2}$$

This is very useful for analyzing the frequency spectrum of a signal, but it has a crucial downside: it eliminates all time information. That is, even though it gives us a perfect view of what frequencies are present in the signal, we have no information about when those frequencies occur. In order to remedy this, we use the Gábor transform, also called a short-time Fourier transform, defined by:

$$\tilde{f}_g(t,\omega) = \int_{\mathbb{R}} f(\tau)\overline{g}(\tau - t)e^{-i\omega t}d\tau \tag{3}$$

The function $\overline{g}(\tau - t)$ acts as a filter in the time domain that localizes the signal over a window centered at time τ. By integrating with respect to τ, we slide this filter down the entire time domain to extract frequency information at each point in time. We visualize this information in a spectrogram. Although we cannot have perfect time resolution and frequency resolution simultaneously, since when we increase time resolution we lose frequency information and vice versa, we can adjust parameters to increase resolution in either direction as needed. Additionally, different Gábor windows have different properties, such as sensitivity to changes in parameters. A simple Gábor window that we will use to reproduce the music score is a Gaussian:

$$g(t) = e^{-a(t-\tau)^2} \tag{4}$$

Where the window width is determined by a. We will also use the Mexican hat wavelet, defined as:

$$\psi(t) = \frac{2}{\sqrt{3\sigma}\pi^{\frac{1}{4}}}(1 - (\frac{t}{\sigma})^2)e^{-t^2}2\sigma^2 \tag{5}$$

Where the window width is determined by σ. We will also use a square wave known as a Shannon window.

3 Algorithm Implementation and Development

First, we load the data and establish our domain. We remember to scale our frequency domain by $\frac{2\pi}{L}$, since the FFT assumes 2π-periodic signals. Then for each spectrogram we create, we create a vector tslide that contains all of our values of τ. Then we iterate across all values of tslide, inside which we initialize our Gábor window (which is shifted by tslide(j), where j is the index of the for loop), multiply the signal by the filter, take the **fft** of the localized signal, and add this to a matrix which stores all of the values of the spectrogram, using **fftshift** to undue the shifting that the function **fft** introduces. We then divide by 2π to obtain accurate frequencies in Hertz and plot the data using **pcolor**.

4 Computational Results

4.1 Part One

We see that a reasonably good window thickness corresponds to roughly $a = 85$ for the Gaussian filter, and incrementing tslide by about .3 seconds corresponds to a reasonably good sampling rate. When we oversample we produce a slightly clearer spectrogram but runtime severely increases, and undersampling causes us to lose a lot of information in both the time and frequence domain. A thicker window causes us to lose time information, and a narrower window causes us to lose frequency information. All Gábor windows performed similarly- the Shannon window had more frequency resolution and less time resolution, while the Mexican Hat wavelet has more time resolution and less frequency resolution (both by only slight amounts).

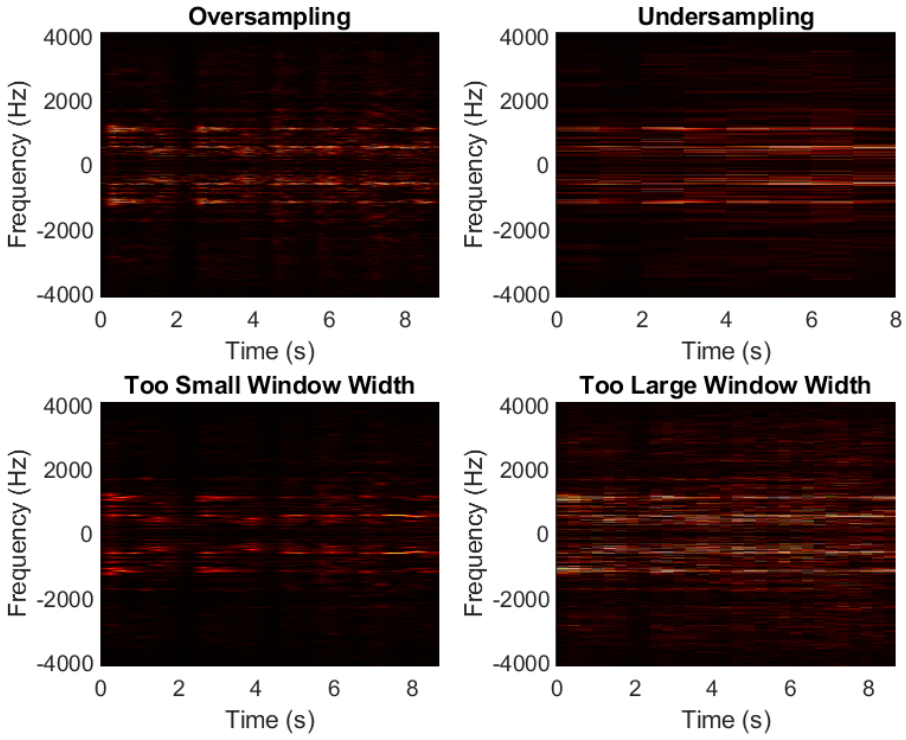

Figure 1: These four spectrograms show what happens when we manipulate the sampling rate (oversampling: top left, undersampling: top right) and window width (too small: bottom left, too large: bottom right) of our Gábor transforms, when the Gábor window is a Gaussian.

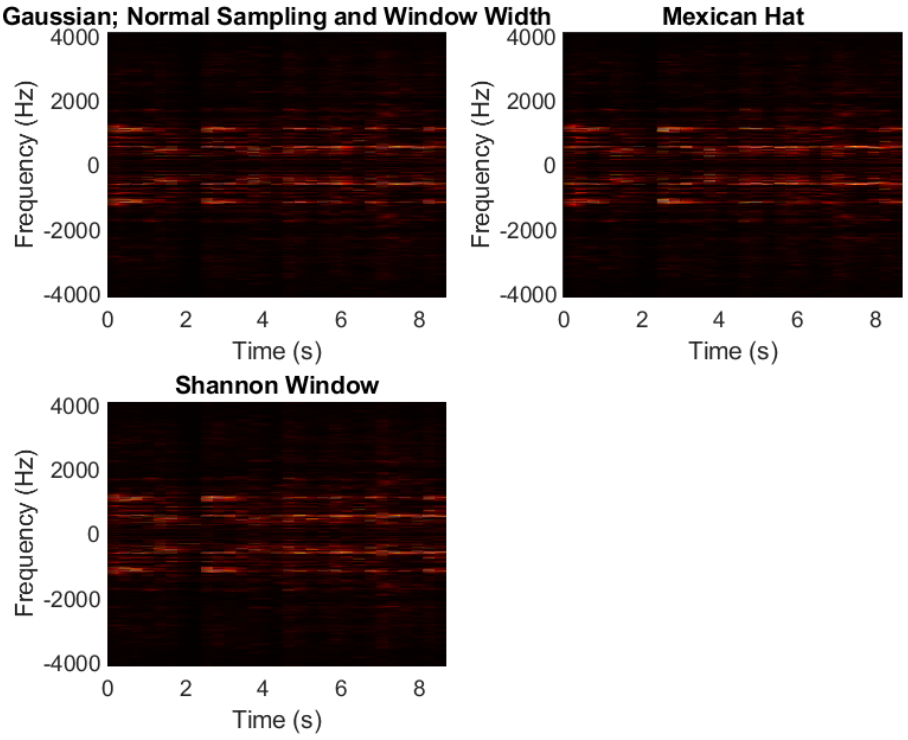

Figure 2: These three spectrograms show what happens when we change our Gábor window to be a Gaussian (top left), Mexican hat wavelet (top right), or Shannon window (bottom left), with suitable chosen and roughly equal sampling rates and window widths.

4.2 Part Two

In Figure 3 shown below, we have reproduced both music scores, as desired. This score excludes overtones, which are integer multiples of the fundamental frequency. We notice that while both scores seem relatively clean, there are more residual frequencies around the fundamental frequency at each point in time in the piano score, which is the essential timbral difference between the recorder and the piano.

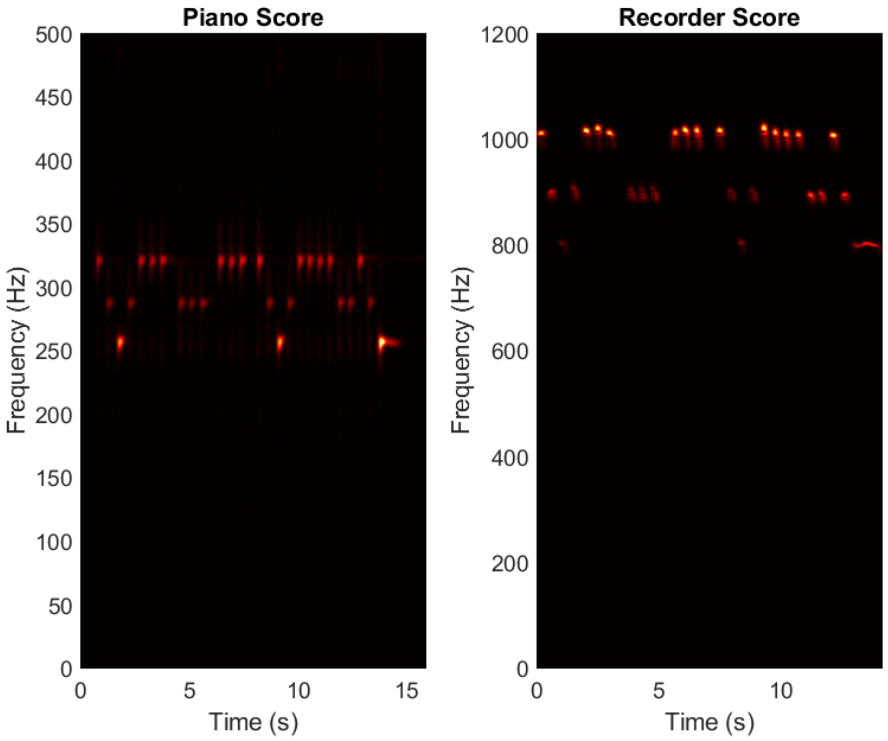

Figure 3: These are the scores for 'Mary Had a Little Lamb' played on piano (left) and recorder (right) with overtones removed. To read off the notes, we can convert the frequencies to musical notes.

5 Summary and Conclusions

Starting from only information in the time domain, we analyzed three audio signals using time-frequency analysis. We did this by using the Gábor transform, and explored how varying window width, sampling rate, and the filter itself affected the resulting spectrogram. We then

took the Gábor transform of the piano and recorder audio signals and excluded overtones to produce the scores for both clips of music.

Appendix A: Function Descriptions

- abs: 2-norm
- exp: "e to the"
- fft: Fast Fourier Transform
- fftshift: interchanges first and second halves of the domain
- linspace: creates vector with linearly spaced points
- print: saves figure
- title: titles plot
- xlabel: labels x-axis on plot
- zeros: creates matrix of zeros

Appendix B: Code

```
clear; clc; close all;

%Part 1
load handel
v = y';
t = (1:length(v))/Fs;
%plot(t,v);
%xlabel('Time [sec]');
%ylabel('Amplitude');
%title('Signal of Interest, v(n)');
%p8 = audioplayer(v,Fs);
%playblocking(p8);
fs = 8192;
L = length(v) / fs;
k=(1/L)*[0:(length(v)-1)/2 -(length(v)-1)/2:-1];
    ks=fftshift(k);

%Gaussian
tslide = 0:.05:L;
a = 50;
Sgt_spec = zeros(length(tslide),length(v));
for j = 1:length(tslide)
```

```
22      g = exp(-a*(t-tslide(j)).^2);
23      Sg = g.*v;
24      Sgt = fft(Sg);
25      Sgt_spec(j,:) = abs(fftshift(Sgt)); % We don't
            want to scale it
26 end
27 figure(1)
28 subplot(2,2,1);
29 pcolor(tslide, ks, Sgt_spec.'),
30 shading interp
31 %set(gca,'Ylim',[-50 50],'Fontsize',16)
32 colormap(hot)
33 xlabel('Time (s)');
34 ylabel('Frequency (Hz)');
35 title('Oversampling');
36
37 tslide = 0:1:L;
38 a = 50;
39 Sgt_spec = zeros(length(tslide),length(v));
40 for j = 1:length(tslide)
41      g = exp(-a*(t-tslide(j)).^2);
42      Sg = g.*v;
43      Sgt = fft(Sg);
44      Sgt_spec(j,:) = abs(fftshift(Sgt)); % We don't
            want to scale it
45 end
46 subplot(2,2,2);
47 pcolor(tslide, ks, Sgt_spec.'),
48 shading interp
49 %set(gca,'Ylim',[-50 50],'Fontsize',16)
50 colormap(hot)
51 xlabel('Time (s)');
52 ylabel('Frequency (Hz)');
53 title('Undersampling');
54
55 tslide = 0:.3:L;
56 a = 2000;
57 Sgt_spec = zeros(length(tslide),length(v));
58 for j = 1:length(tslide)
59      g = exp(-a*(t-tslide(j)).^2);
60      Sg = g.*v;
61      Sgt = fft(Sg);
62      Sgt_spec(j,:) = abs(fftshift(Sgt)); % We don't
```

```
        want to scale it
end
subplot(2,2,3);
pcolor(tslide, ks, Sgt_spec.'),
shading interp
%set(gca,'Ylim',[-50 50],'Fontsize',16)
colormap(hot)
xlabel('Time (s)');
ylabel('Frequency (Hz)');
title('Too Small Window Width');

tslide = 0:.3:L;
a = 10;
Sgt_spec = zeros(length(tslide),length(v));
for j = 1:length(tslide)
    g = exp(-a*(t-tslide(j)).^2);
    Sg = g.*v;
    Sgt = fft(Sg);
    Sgt_spec(j,:) = abs(fftshift(Sgt)); % We don't
        want to scale it
end
subplot(2,2,4);
pcolor(tslide, ks, Sgt_spec.'),
shading interp
%set(gca,'Ylim',[-50 50],'Fontsize',16)
colormap(hot)
xlabel('Time (s)');
ylabel('Frequency (Hz)');
title('Too Large Window Width');
print(gcf,'-dpng','Explore_Spectrogram.png');

tslide = 0:.3:L;
a = 85;
Sgt_spec = zeros(length(tslide),length(v));
for j = 1:length(tslide)
    g = exp(-a*(t-tslide(j)).^2);
    Sg = g.*v;
    Sgt = fft(Sg);
    Sgt_spec(j,:) = abs(fftshift(Sgt)); % We don't
        want to scale it
end
figure(2);
subplot(2,2,1);
```

```
pcolor(tslide, ks, Sgt_spec.'),
shading interp
%set(gca,'Ylim',[-50 50],'Fontsize',16)
colormap(hot)
xlabel('Time (s)');
ylabel('Frequency (Hz)');
title('Gaussian; Normal Sampling and Window Width');

%Mexican Hat
tslidem = 0:.3:L;
om = .05;
Smt_spec = zeros(length(tslidem),length(v));
for j = 1:length(tslidem)
    m = 2/(sqrt(3*om)*pi^.25)*(1-((t-tslidem(j))/om).^2).*exp(-((t-tslidem(j))/om).^2/2);
    Sm = m.*v;
    Smt = fft(Sm);
    Smt_spec(j,:) = abs(fftshift(Smt)); % We don't want to scale it
end
subplot(2,2,2);
pcolor(tslidem, ks, Smt_spec.'),
shading interp
%set(gca,'Ylim',[-50 50],'Fontsize',16)
colormap(hot)
xlabel('Time (s)');
ylabel('Frequency (Hz)');
title('Mexican Hat');

%Shannon
tslides = 0:.3:L;
width = .2;
Sst_spec = zeros(length(tslides),length(v));
for j = 1:length(tslides)
    s = abs(t - tslides(j)) <= width / 2;
    Ssh = s.*v;
    Sst = fft(Ssh);
    Sst_spec(j,:) = abs(fftshift(Sst)); % We don't want to scale it
end
subplot(2,2,3);
pcolor(tslides, ks, Sst_spec.'),
```

```
142 shading interp
143 %set(gca,'Ylim',[-50 50],'Fontsize',16)
144 colormap(hot)
145 xlabel('Time (s)');
146 ylabel('Frequency (Hz)');
147 title('Shannon Window');
148 print(gcf,'-dpng','Explore_Wavelets.png');
149
150 %Part 2
151 %Piano
152 [yp,Fsp] = audioread('music1.wav');
153 tr_piano=length(yp)/Fsp; % record time in seconds
154 %figure(4)
155 %plot((1:length(yp))/Fsp,yp);
156 %xlabel('Time [sec]'); ylabel('Amplitude');
157 %title('Mary had a little lamb (piano)');
158 %p8 = audioplayer(yp,Fsp); playblocking(p8);
159 kp=(1/tr_piano)*[0:(length(yp))/2-1 -(length(yp))
      /2:-1]; kps=fftshift(kp);
160 tp = (1:length(yp)) / Fsp;
161 tslidep = 0:.1:tr_piano;
162 ap = 100;
163 Sgt_spec_p = zeros(length(tslidep),length(yp));
164 for j = 1:length(tslidep)
165     g = exp(-ap*(tp-tslidep(j)).^2);
166     Sg_p = g.*yp';
167     Sgt_p = fft(Sg_p);
168     Sgt_spec_p(j,:) = abs(fftshift(Sgt_p)); % We
          don't want to scale it
169 end
170 figure(3)
171 subplot(1,2,1);
172 pcolor(tslidep, kps, Sgt_spec_p.'),
173 shading interp
174 ylim([0,500]);
175 %set(gca,'Ylim',[-50 50],'Fontsize',16)
176 colormap(hot)
177 xlabel('Time (s)');
178 ylabel('Frequency (Hz)');
179 title('Piano Score');
180
181 %Recorder
182 [yr,Fsr] = audioread('music2.wav');
```

```
tr_rec=length(yr)/Fsr; % record time in seconds
%plot((1:length(yr))/Fsr,yr);
%xlabel('Time [sec]'); ylabel('Amplitude');
%title('Mary had a little lamb (recorder)');
%p8 = audioplayer(yr,Fsr); playblocking(p8);
kr=(1/tr_rec)*[0:(length(yr))/2-1 -(length(yr))
    /2:-1]; krs=fftshift(kr);
tr = (1:length(yr)) / Fsr;
tslider = 0:.1:tr_rec;
ar = 100;
Sgt_spec_r = zeros(length(tslider),length(yr));
for j = 1:length(tslider)
    g = exp(-ar*(tr-tslider(j)).^2);
    Sg_r = g.*yr';
    Sgt_r = fft(Sg_r);
    Sgt_spec_r(j,:) = abs(fftshift(Sgt_r)); % We
        don't want to scale it
end
subplot(1,2,2);
pcolor(tslider, krs, Sgt_spec_r.'),
shading interp
ylim([0,1200]);
%set(gca,'Ylim',[-50 50],'Fontsize',16)
colormap(hot)
xlabel('Time (s)');
ylabel('Frequency (Hz)');
title('Recorder Score');
print(gcf,'-dpng','Mary_Score.png');
```