AMATH 482 HW 2

The Use of Gabor Transforms in Analyzing Time-varying Signals

Zachary McNulty
zmcnulty, ID: 1636402

Abstract: This paper aims to demonstrate the usefulness of the Gabor transform in extracting frequency information from time-varying signals. As many signals vary in time, this technique has a wide range of applications including image analysis, computational neuroscience, and even the decomposition of whale songs. It will discuss the trade-offs associated to the transform and some of the options there are for defining a sampling window. As a case study, we will use this technique to study a few pieces of music.

University of Washington
February 2019

1 Introduction and Overview

Many of the signals we come into contact with are time-varying. Often the relative order of these frequencies carries information in and of itself and thus the local frequency content at each point in time is important: take a song or speech, scramble it up, and suddenly it loses all meaning. However, the Fourier Transform only gives us a global measurement of the frequency content, and thus this meaning is lost. The Gabor Transform seeks to improve upon this short-coming of the Fourier Transform. By only looking a specific window of time, it can break the signal down into several pieces, separating the frequency contents at each point in time. As we will see, this comes with a trade-off: we lose frequency resolution.

In this paper, we use this technique to decompose several pieces of music. The first is Handel's Messiah which we will use to study the properties of different Gabor windows. The other two pieces are *Mary Had a Little Lamb* played on a piano and recorder separately. Using what we learned about Gabor windows in the first part, we will try to make inferences about the properties of the given instruments and the sounds they generate. Using this information, we will try to recreate the music score used to produce these signals.

2 Theoretical Background

The Fourier Transform is the primary tool for converting between the spatial and frequency domains of a signal. Below is the transform and its inverse:

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

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

Where k is the wavenumber. It may not be clear from **Equations 1** or **2** why this encodes frequency information, but expanding e^{-ikx} provides some insight:

$$e^{-ikx} = \cos(kx) - i\sin(kx) \tag{3}$$

Equation 3 makes it clear why then k represents the wavenumber. Thus the transform is essentially taking the inner product of our function with a bunch of periodic sines and cosines where, roughly speaking, frequencies/wavenumbers more similar to our signal get assigned higher values. In this way, spatial information is converted to the frequency domain. However, as mentioned earlier this transformation loses all spatial information in order to get complete frequency resolution, but that information can sometimes be really important. The Gabor Transform is a compromise:

$$\mathcal{G}[f](t,\omega) = \int_{-\infty}^{\infty} f(\tau) g(\tau - t) e^{-i\omega\tau} d\tau \tag{4}$$

Where $g(\tau - t)$ defines a window centered at time t. This window acts as a filter, allowing us to focus our frequency analysis on a specific window in time. As we slide t across the temporal domain, we get a picture of the frequencies at each point in time. While this gives us information on where in time each of these frequencies are occurring (increased time resolution), the width of the window limits the wavelengths of frequencies we can detect. Thus, we lose frequency resolution. However, expand the window and we lose time resolution, and thus there is always this trade-off between spatial and frequency resolution, as dictated by the Heisenberg Uncertainty Principle. Because of this trade-off, it is important we choose an appropriate window. The width of the window should be chosen based on the range of frequencies of interest: we want them all to fit in the window. Some of the potential/common window functions are shown in **Figure 1**. Their equations, with (inverse) width parameter a and centered at t are:

Gaussian: $g(\tau - t) = e^{-a(\tau - t)^2}$

Super Gaussian: $g(\tau - t) = e^{-a(\tau - t)^{10}}$

Mexican Hat Wavelet: $g(\tau - t) = (1 - (\tau - t)^2) e^{-a(\tau - t)^2/2}$

Shannon (Step-Function):

$$g(\tau - t) = \begin{cases} 1 & x \in [t - \frac{1}{2a}, t + \frac{1}{2a}] \\ 0 & otherwise \end{cases}$$

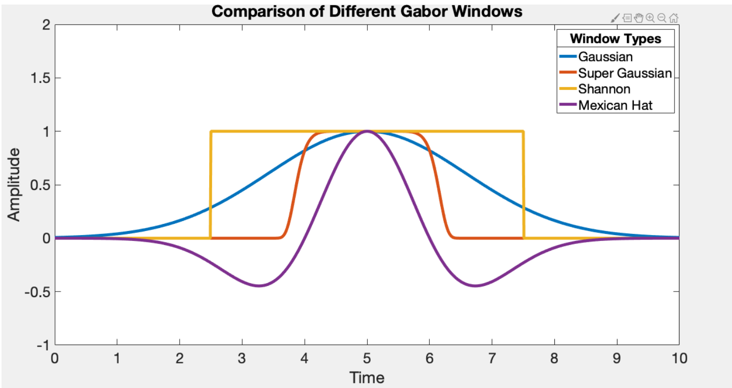


Figure 1: A few different Gabor Windows, each with the same width parameter.

These windows decay at different rates and have their own specific properties, so the window should be carefully chosen to fit the signal and task at hand. When trying to visualize the results of a Gabor Transform, spectrograms are useful tools. These are plots of frequency versus time, where color intensity highlights the prevalence of a given frequency at a specific time.

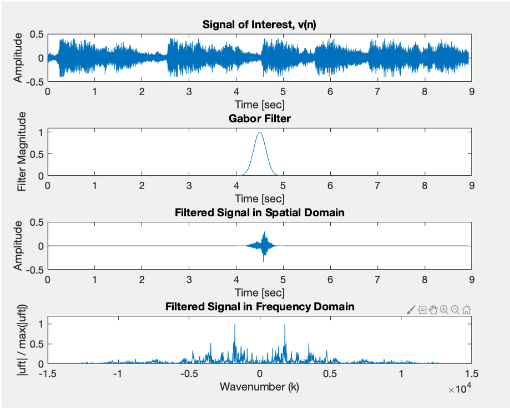
3 Algorithm Implementation and Development

3.1 Analysis of Gabor Windows and Handel's Messiah

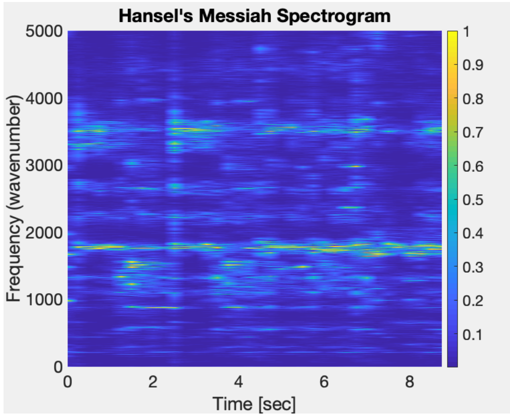
We begin by loading a sample of Handel's music as a simple vector of amplitudes. Using the sampling rate Fs, we can reconstruct the time domain for this signal. We strip a single data point from the signal v to give it an even length. This simplification allows v to be compatible with the wavenumbers we calculate (which have an even length) and has a negligible effect on our overall data. Given the time domain we found, we reconstructed our wavenumbers, shifting them to be centered at zero (Appendix B, code 0-28). Our goal is the use a Gabor Transform to analyze the frequency content of this music as a function of time. First, we define some of the properties of our window: its sampling rate $tstep$ (how much it slides each iteration), its width which is inversely related to a, and its function (i.e. gaussian versus Shannon). Unless otherwise specified, we used a gaussian window with width $a = 30$ and sampling rate $tstep = 0.25$ (Appendix B, code 29-52). For each sampling frame in our time domain, we plot the original signal, plot our Gabor window, apply the filter to the signal in the time domain, use the Fourier Transform to convert to the frequency domain, and plot the result in the frequency domain using our pre-calculated wavenumbers (Appendix B, code 53-98). In each sampling frame, we store the resulting frequency information. Lastly, we plot this frequency data at each point of time in a spectrogram (Appendix B, code 100 - 112). We repeat this process several times, varying the properties of the window.

3.2 Exploring Instrumental Differences: Piano vs. Recorder

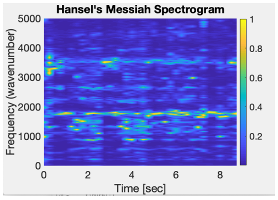
Again, the signal produced by these instruments is time varying so to understand its time-dependent behavior, we aim to perform a Gabor Transform. After loading in our signal, we began this stage by repeating what we did in the previous: applying a Gabor Transform and playing with the properties of our Gabor window to find what works best (Appendix B, code 120 - 196). In the end, we decided to use a Shannon window with width parameter $a = 3$ and $tstep = 0.25$,. Additionally, at each window sampling frame we store the most prevalent frequencies. We do this by finding the index of the maximum magnitude in our frequency domain and mapping this index onto our wavenumbers. As our signal is in the time domain, these wavenumbers represent angular frequency. Dividing them by 2π converts them to Hz (Appendix B, code 199; 208). Once we find the most prevalent frequency at each time point, we simply find which note this frequency is most closely associated with (Appendix B, code: 212 - 233). To help visualize our results and spot some properties of the instrument (i.e. the overtones) we plot the spectrogram data we collected earlier (Appendix B, code: 237 - 248). We then repeat this entire process all over again for the recorder's signal (Appendix B, code: 256 - 366).



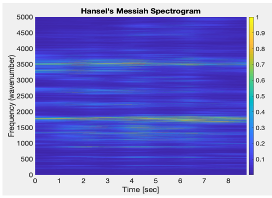
(a) Filtering Signal with Gabor Window



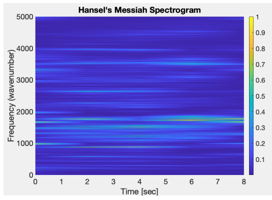
(b) Standard Spectrogram



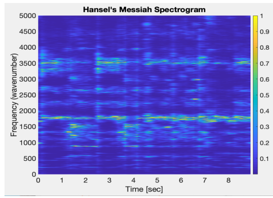
(c) Narrow Window



(d) Wide Window



(e) Undersampling



(f) Oversampling

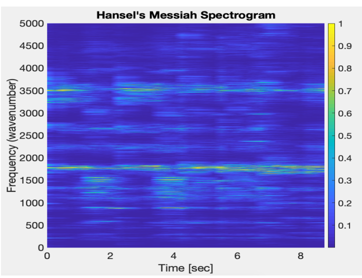
Figure 2: Analyzing effects window properties have on the Gabor Transform with a gaussian filter

4 Computational Results

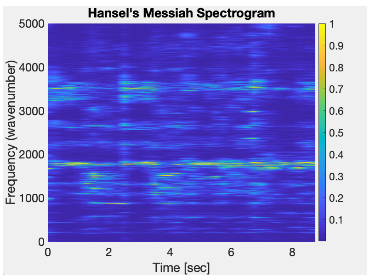
4.1 Analysis of Gabor Windows and Handel's Messiah

We initially began by exploring how various properties of the window would effect the outcome, displaying these results in **Figure 2**. **2a** shows what a typical analysis would look like: we start with some signal, run the signal through our filter, get a filtered signal, and then take the Fourier Transform of that to get the prevalent frequencies. In **2b**, we show what the spectrogram output of this process would look like with middle of the road parameters. We will use this as a benchmark for comparison. We can see that there are two distinct bands of prominent frequency around wavenumber 1800 and 3500, but there is a fair bit of temporal variation in the frequency content outside of that.

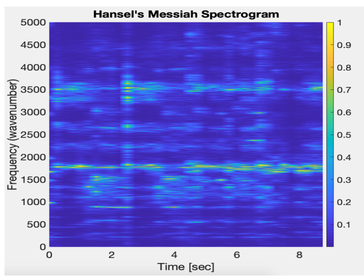
When we narrow the window ($a = 1000$), as we did in **Figure 2c**, we get a really good look at the temporal variation in the frequency content. Normally, we would lose track of our long wavelength signals (low frequency/wavenumber), but here those frequencies do not seem too prominent so that is not much of an issue. We can see there are times, like around 2 and 4 seconds, where the frequency content is much richer than other times. However, this reduction in frequency resolution makes it more difficult to differentiate long-term patterns and to determine individual frequencies (blurred areas of activity versus fine lines). When we widen the window ($a = 1$) in **2d**, we gain frequency resolution at the cost of temporal resolution. This allows us to look for a wide range of frequencies and makes the persistent 1800/3500 wavenumbers more noticeable, but we lose a lot of



(a) Super Gaussian



(b) Mexican Hat



(c) Shannon

Figure 3: Analyzing the effect of different window functions on the Gabor Transform

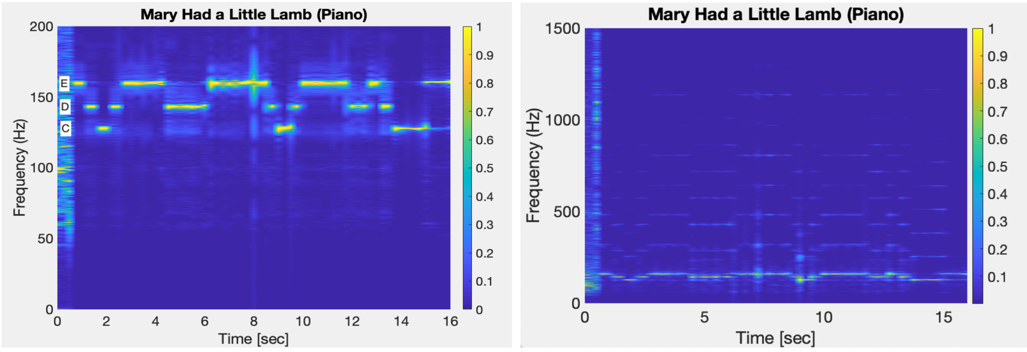


Figure 4: Piano at two different frequency ranges: left shows fundamental frequencies while the right helps show all the related overtones.

the temporal variation we observed earlier which would be important in music. We see this same issue in **2e** where we lowered the sampling rate ($tstep = 2$). When we increase the sampling rate ($tstep = 0.01$) in **2f**, we regain these areas of rich frequency content and have fairly fine areas of prominent frequencies, but there are also some times where the frequency patterns are very erratic. With such a small timestep, we are more likely to accidentally measure a rest period within the music and that is hardly what we are interested in. Furthermore, this took a considerable amount of time to run.

When we experimented with the window function, there did not seem to be too much of a difference. In **Figure 3**, we can see that the super gaussian window gave a bit clearer of a spectrogram while the Mexican Hat and Shannon windows seem to capture the areas of rich frequency content a bit better. The flanking negative areas of the Mexican Hat window may make it better at picking up changes in frequency content, but in this context all window functions seem to perform adequately.

4.2 Exploring Instrumental Differences: Piano vs. Recorder

We chose to reconstruct the music score for *Mary Had a Little Lamb* from the piano as we felt it likely had a smaller margin of error: the piano is a bit more precise of an instrument than the recorder. **Figure 4** shows the resulting spectrograms. In the left image, we can clearly see the dominant frequencies jump between 3 distinct levels: E (164.81 Hz), D (146.83 Hz), and C (130.81 Hz) from the top down. Sampling the note every ≈ 0.5 seconds yields the following music score:

Mar-	-y	had	a	lit-	-tle	lamb,	litt-	le	lamb,	lit-	-tle	lamb.
E	D	C	D	E	E	E	D	D	D	E	E	E

Mar-	-y	had	a	lit-	-tle	lamb	whose	fleece	was	white	as	snow.
E	D	C	D	E	E	E	E	D	D	E	D	C

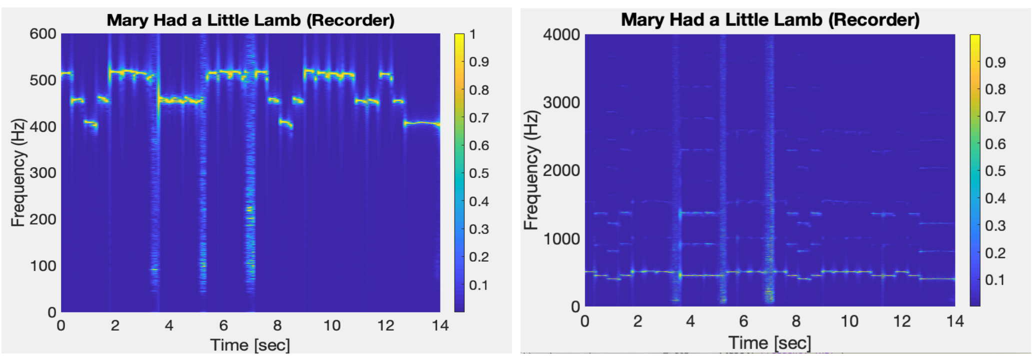


Figure 5: Recorder at two different frequency ranges: left shows fundamental frequencies while the right helps show all the related overtones.

In the right plot of **Figure 4**, we can observe several layers of overtones above the fundamental frequencies mentioned earlier - the faint bands echoing the pattern of the fundamental notes but at higher frequencies. These are a fundamental property of the instrument itself. It appears the overtones are prominent up until around the forth or fifth harmonic before disappearing. In **Figure 5**, we see the same two plots but for the recorder. In the left plot, we see the same relative trend in frequencies, moving between three frequency levels. These frequencies are playing the same three notes – E,D,C – but at a higher octave. However, these frequencies are much choppier. In comparison to the piano, there is much more variation around each of these three frequency levels, blurring the spectrogram. This suggests the recorder is a less precise instrument than the piano: it is harder to maintain a consistent note. Furthermore, observing the right plot of **Figure 5**, we can see the recorder has much fewer overtones. This could be why the noise produced by the recorder is not as rich as that produced by the piano.

5 Summary and Conclusions

The Fourier Transform is a powerful tool for analyzing the frequency content of a signal, but because it maps from the spatial domain to the frequency domain, all spatial information is lost. This is problematic when the signal is time-varying as often the relative order of frequencies is important in and of itself. As a compromise to this, the Gabor Transform is a tool that can be used to gain some spatial resolution, but it comes at the cost of frequency resolution. There is no way to avoid this trade-off between frequency and spatial resolution so it is crucial to carefully consider how to apply the Gabor Transform. Most prominently, it is important to choose a proper window, deciding what window function, width, and sampling rate fits the task at hand best. This paper highlighted the different roles each of these window parameters have. Using these ideas, we were able to successfully reconstruct the music score of *Mary Had a Little Lamb* from a time-varying signal (a piano). Furthermore, taking the signal from playing this same music score on a recorder, we were able to make inferences about differences between these two instruments and the overtones they produce.

6 Appendix A

Below is a brief summary of the MATLAB functions I used during this project and their functions.

pcolor: Takes a two vectors which define a grid, setting the x and y tick marks, and a 2D matrix. At each point in the grid, it colors the figure based on the corresponding (normalized) value in the 2D matrix. This gives a nice visualization on where the maximum and minimum values of the dataset are occurring. We used it to make the spectrograms.

fftn/ifftn: The former computes the n-dimensional Fourier Transform of the given n-dimensional array. Generates frequency contents in the given n-dimensions. ifftn is its inverse.

fftshift/ifftshift: The former shifts zero frequency component to the center of the frequency spectrum. Works for data in any number of dimensions. Mostly helpful for plotting in the frequency domain, and may not be necessary otherwise. ifftshift is its inverse.

7 Appendix B

```
% AMATH 482 HW 2: Gabor Transforms

close all; clear all; clc

load handel
v = y'/2;
plot((1:length(v))/Fs,v);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');

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

%%
close all; clc;

v = y'/2;
v = v(1:end-1); % remove last point so we have even number of points
L = 9; % tspan(end); length of our signal in time (or space) domain


tspan = (1:length(v))/Fs;
n = length(v); % ALWAYS has to be even

% rescale [-pi pi] domain fft assumes to fit our domain [-L, L]
k = 2*pi / (2*L) .* [0:(n/2 - 1) -n/2:-1];
ks = fftshift(k);

tstep = 0.25; % how much to slide the window after each measurement to
    new measurement
a = 3; %width parameter of our gabor filter; actually width is prop to
    1/a

% list of values to center filter at; collect frequencies for each
% one of the windows centered at this point for spectrogram
tau = 0:tstep:tspan(end);

spectrogram_data = zeros(n, length(tau));

figure(1)

for j = 1:length(tau)
    % Gaussian Window
    %gabor_filter = exp(-a*(tspan - tau(j)).^2);

    % Super Gaussian Window
    %gabor_filter = exp(-a*(tspan - tau(j)).^10);

    % Mexican Hat Window
    %gabor_filter = (1-(tspan - tau(j)).^2).*exp(-a*(tspan - tau(j)).^2
        ./ 2);

    % Shannon (Step-function) Window
    gabor_filter = abs(tspan - tau(j)) <= 1/(2*a);

    % plot our original signal
    subplot(411)
```

```
    plot((1:length(v))/Fs,v);
    ylim([-0.5,0.5])
    ylabel('Amplitude');
    xlabel('Time [sec]');
    title('Signal of Interest, v(n)');
    set(gca, 'fontsize', 18);


    % Plot our gabor filter
    subplot(412)
    plot(tspan, gabor_filter);
    %ylim([-1, 2]) % mexican hat wavelet
    ylim([0,1.1]) %gaussian limits
    xlim([0,9])
    title('Gabor Filter')
    ylabel('Filter Magnitude');
    xlabel('Time [sec]');
    set(gca, 'fontsize', 18);


    % plot our filtered signal
    v_filtered = gabor_filter .* v;

    subplot(413)
    plot(tspan, v_filtered);
    xlabel('Time [sec]');
    ylabel('Amplitude');
    title('Filtered Signal in Spatial Domain')
    ylim([-0.5,0.5])
    set(gca, 'fontsize', 18);

    % plot our filtered signals frequency domain
    vft = fft(v_filtered);

    subplot(414)
    plot(ks, abs(fftshift(vft)) ./ max(abs(vft)));
    xlabel('Wavenumber (k)');
    ylabel('|uft| / max(|uft|)');
    title('Filtered Signal in Frequency Domain')
    ylim([0, 1.2])
    set(gca, 'fontsize', 18);

    %drawnow

    spectrogram_data(:, j) = (abs(fftshift(vft)) ./ max(abs(vft)) ).';

end
%%
   % make spectogram plot
   figure(2)
   pcolor(tau, ks, spectrogram_data), shading interp;
   colorbar
   ylim([0, 5000])
   ylabel('Frequency (wavenumber)')
   xlabel('Time [sec]')
   title("Hansel's Messiah Spectrogram")
   set(gca, 'fontsize', 25)
```

```
%% Part 2: Piano
clear all; close all; clc;


% LOAD MUSIC
 figure(1)
 tr_piano=16;  % record time in seconds
 m1 = audioread('music1.wav').';
 Fs1=length(m1)/tr_piano;
 plot((1:length(m1))/Fs1,m1);
 xlabel('Time [sec]'); ylabel('Amplitude');
 title('Mary had a little lamb (piano)');  drawnow
 %p8 = audioplayer(m1,Fs1); playblocking(p8);


 %%

 tspan1 = (1:length(m1))/Fs1;
 L1 = tr_piano;
 n1 = length(m1);
 k1 = 2*pi / (2*L1) .* [0:(n1/2 - 1) -n1/2:-1];
 ks1 = fftshift(k1);
 tstep1 = 0.25; % how much to slide the window after each measurement
     to new measurement
 a1 = 2; %width parameter of our gabor filter; higher a = lower width

% list of values to center filter/gabor window at
tau1 = 0:tstep1:tspan1(end);

% stores the max wavenumber for each gabor window (centered at tau(j))
max_k1 = zeros(1, length(tau1));
spectrogram_data1 = zeros(n1, length(tau1));

for j = 1:length(tau1)
    %gabor_filter1 = exp(-a1*(tspan1 - tau1(j)).^2); % Gaussian
    %gabor_filter1 = exp(-a1*(tspan1 - tau1(j)).^10); % Gaussian

    % Shannon (Step-function) Window
    gabor_filter1 = abs(tspan1 - tau1(j)) <= 1/(2*a1);

    % plot our original signal
    subplot(411)
    plot((1:length(m1))/Fs1,m1);
    xlabel('Time [sec]');
    ylabel('Amplitude');
    title('Mary Had a little lamb (piano)');

    % Plot our gabor filter
    subplot(412)
    plot(tspan1, gabor_filter1);
    %ylim([-2, 3]) % mexican hat wavelet
    ylim([0,1.1]) %gaussian limits

    xlim([0,L1])
    xlabel('Time [sec]');
    title('Gabor Filter')
    ylabel('Filter Magnitude');

    % plot our filtered signal
```

```
    m1_filtered = gabor_filter1 .* m1;

    subplot(413)
    plot(tspan1, m1_filtered);
    xlabel('Time [sec]');
    ylabel('Amplitude');
    title('Filtered Signal in Spatial Domain')
    ylim([-0.7,0.7]);
    xlim([0, L1])


    % plot our filtered signals frequency domain
    m1ft = fft(m1_filtered);

    subplot(414)
    plot(ks1, abs(fftshift(m1ft)) ./ max(abs(m1ft)));
    xlabel('Wavenumber (k)');
    ylabel('|uft| / max(|uft|)');
    title('Filtered Signal in Frequency Domain')
    xlim([-20000,20000])

    % outside "max" call is to filter out the negative wavenumber
    max_k1(j) = max(k1(abs(m1ft) == max(abs(m1ft))));

    spectrogram_data1(:, j) = (abs(fftshift(m1ft)) ./ max(abs(m1ft)) ).';
    %drawnow

end


% convert wavenumber to hertz!
max_freqs1 = max_k1 ./ (2*pi);

%%

notes = ["A", "A#", "B", "C", "C#","D", "D#", "E", "F", "F#", "G", "G#"];
fund_freqs = [27.5, 29.135, 30.863, 32.703, 34.648,36.708, 38.891, 41.203, 43.654,46.249, 48.99, 51.913];

all_notes = [];
all_freqs = [];

for j = 1:8
    for note = notes
        all_notes = [all_notes  strcat(note, num2str(j))];
    end
    for freq = fund_freqs
        % freqs double each time you move up scale
        all_freqs = [all_freqs freq*(2^(j-1))];
    end
end

music_score1 = [];

for freq = max_freqs1
    [min_val, index] = min(abs(freq - all_freqs));
    music_score1 = [music_score1 all_notes(index)];
end

```

```
%% make spectogram plot for Piano!
close all; clc;

figure(3)
% divide by 2pi to convert wavenumber to Hz
pcolor(tau1, ks1./(2*pi), spectrogram_data1), shading interp;
colorbar
ylim([0, 1000]) % to see overtones
%ylim([0, 200]) % to see base notes
ylabel('Frequency (Hz)')
xlabel('Time [sec]')
title('Mary Had a Little Lamb (Piano)')
set(gca, 'fontsize', 25);



%% Part 2: Recorder
clearvars -except all_notes all_freqs; close all; clc;


 figure(2)
 tr_rec=14;  % record time in seconds
 m2=audioread('music2.wav').';
 Fs2=length(m2)/tr_rec;
 plot((1:length(m2))/Fs2,m2);
 xlabel('Time [sec]'); ylabel('Amplitude');
 title('Mary had a little lamb (recorder)');
 p8 = audioplayer(m2,Fs2); playblocking(p8);

 %%

 tspan2 = (1:length(m2))/Fs2;
 L2 = tr_rec;
 n2 = length(m2);
 k2 = 2*pi / (2*L2) .* [0:(n2/2 - 1) -n2/2:-1];
 ks2 = fftshift(k2);
 tstep2 = 0.5; % how much to slide the window after each measurement to
     new measurement
 a2 = 2; %width parameter of our gabor filter

% list of values to center filter at; collect frequencies for each
% one of the windows centered at this point for spectrogram
tau2 = 0:tstep2:tspan2(end);

% stores the max wavenumber for each gabor window (centered at tau(j))
max_k2 = zeros(1, length(tau2));
spectrogram_data2 = zeros(n2, length(tau2));

for j = 1:length(tau2)
    % Shannon (Step-function) Window
    gabor_filter2 = abs(tspan2 - tau2(j)) <= 1/(2*a2);

    % plot our original signal
    subplot(411)
    plot((1:length(m2))/Fs2,m2);
    xlabel('Time [sec]');
    ylabel('Amplitude');
    title('Mary Had a little lamb (recorder)');

    % Plot our gabor filter
```

```
    subplot(412)
    plot(tspan2, gabor_filter2);
    ylim([0,1.1]) %gaussian limits

    xlim([0,L2])
    xlabel('Time [sec]');
    title('Gabor Filter')
    ylabel('Filter Magnitude');

    % plot our filtered signal
    m2_filtered = gabor_filter2 .* m2;

    subplot(413)
    plot(tspan2, m2_filtered);
    xlabel('Time [sec]');
    ylabel('Amplitude');
    title('Filtered Signal in Spatial Domain')
    ylim([-0.7,0.7]);
    xlim([0, L2])


    % plot our filtered signals frequency domain
    m2ft = fft(m2_filtered);

    % filter overtones?

    subplot(414)
    plot(ks2, abs(fftshift(m2ft)) ./ max(abs(m2ft)));
    xlabel('Wavenumber (k)');
    ylabel('|uft| / max(|uft|)');
    title('Filtered Signal in Frequency Domain')
    xlim([-20000,20000])

    % outside "max" call is to filter out the negative wavenumber
    max_k2(j) = max(k2(abs(m2ft) == max(abs(m2ft))));

    spectrogram_data2(:, j) = (abs(fftshift(m2ft)) ./ max(abs(m2ft)) )
        .';

    drawnow

end



% convert wavenumber to hertz!
% dividing by an extra pi makes it exactly what I would expect?

max_freqs2 = max_k2 ./ (2*pi);

%% Find Music Score!
music_score2 = [];

for freq = max_freqs2
    [min_val, index] = min(abs(freq - all_freqs));
    music_score2 = [music_score2 all_notes(index)];
end


%% make spectogram plot for Recorder!
close all; clc;
```

```
figure(4)

% divide by 2pi to convert wavenumber to Hz
pcolor(tau2, ks2./(2*pi), spectrogram_data2), shading interp;
colorbar
hold on;

ylim([0, 1000]) % to see overtones
%ylim([0, 200]) % to see base notes
ylabel('Frequency (Hz)')
xlabel('Time [sec]')
```