Time-Frequency Analysis: Applications in Music

Constance Wang

Abstract

Time-frequency analysis is utilized to study different pieces of music. This type of analysis is particularly useful in the study of music since the music signal changes frequency character across time. Using Gabor transforms and spectrograms, both these tools and the music pieces can be studied. These tools can capture and visually indicate the changes in frequency across length of the song, which can be used to study the actual music notes played and the characteristics of the instrument(s) in the song.

Section I – Introduction and Overview

Time-frequency analysis is a technique that is used to determine how frequencies change over time. In this problem, we are analyzing different music files using time-frequency analysis techniques, specifically using the Gabor transform to construct spectrograms. A musician would describe music as the composition of different music notes (melodies) that are played according to some rhythm. A technical view of music is that it is an audio signal that has frequencies (music notes) that change over time (rhythms). This means that music lends itself well to time-frequency analysis.

In the first part of the problem, exploration of Gabor filtering and spectrograms will help us better understand how different parameters of the Gabor transform can change the resulting spectrograms. In the second part of the problem, two different versions (two instruments) of "Mary had a Little Lamb" have been recorded. The Gabor transform will be used to determine the frequencies (music notes) played over the course of the song to reproduce the music score. The presence of overtones in an instrument can also be seen with these analysis techniques.

Section II – Theoretical Background

In the previous problem solved with Fluffy the dog, Fourier analysis used ultrasound signals to locate a marble. Since the nature of the marble is constant in time, its frequency character of the marble is constant in time (only its location changes). In this problem, however, the frequency character of music changes in time since different music notes are played across time.

Gabor Transform

Fourier analysis is an extremely powerful tool to decompose a signal into different frequencies, so to analyze frequencies across time, the idea is to take Fourier transforms at different points in time. However, choosing arbitrary points in time can pose problems such as Gibbs Phenomenon as well as any human error in choosing different points in time. The solution is to use a function (such as the Gaussian filter function) to look at signals at a particular window in time, and shift this window across the entire time domain. This is the general idea behind the Gabor transform (Equation 1), reproduced below:

$$\tilde{f}(\tau,\omega) = \int_{-\infty}^{\infty} f(t)g(t-\tau)e^{-ikt}dt \qquad \text{Gabor Transform (1)}$$

In comparison to the Fourier transform, the only difference is the $g(t-\tau)$ term added to the original definition. This term is the window function, which has some center at τ. This window function should essentially zeros out the signal outside of the window. The transformed signal also depends on τ, which is also the location in time at which the signal is analyzed. In the following

solutions, a Gaussian function is used, but in general, characteristics of this function are: $g(t-\tau)$ is real, symmetry, and its norm is 1. The Gaussian is shown in Equation 2 below:

$$windowfunction = e^{-a((t-\tau)^2)}$$ Gaussian Function (2)

Just like in filtering for time-independent signals, there is some width associated with the window function. For the Gabor transform, the width of the window function will determine what kind of information is given by the transform. At the two extremes, an infinite and zero width window, all frequency or time (respectively) information is captured. To understand this concept, consider an infinite width window. This would be like assuming the signal occurred at a single point in time (no time information captured), much like taking a simple Fourier transform. For a narrow window, it is difficult to capture frequency information, since frequency information needs to occurs across some expanse of time. An intermediate window width would capture some time frequency information and some time information (typically what is used).

Gabor Transform in Time-Frequency Analysis

To best utilize the Gabor transforms, the transforms must be taken at multiple, consecutive instances in time. Visually, this would look like "sliding" the transform across time, taking Gabor transforms at different τ. The distance between neighboring τ values taken, or the $\Delta\tau$, must be taken as to prevent undersampling (where there is little to no overlap between neighboring filter functions) but avoid excessive oversampling (where there is overlap). Oversampling can be a problem because spectrograms will take a long time to display with more values of τ.

Spectrograms

Recording a series of consecutive Gabor transforms will give information about the frequency content at a particular τ. Each of these Gabor transforms can be visualized in a visual plot, where the x-axis is time and the y-axis is frequency. All Gabor transforms taken across time are recorded visually in this plot. The magnitude of frequency at a specific time will be a certain "color" on this visual plot, according to some color bar. This visual plot is known as a spectrogram. Spectrograms are extremely useful, as we will see in the second part of the assignment to best reproduce the music score (without too much knowledge about how to read/write music).

Section III – Algorithm Implementation and Development

I split the Matlab code into two parts, one for each problem.

Part I

In this part, the main goal is to explore time-frequency analysis techniques, specifically the Gabor transform and spectrograms. To start, initialization of variables must occur. The signal used is a built-in Matlab song by Handel. When loading this in, the sample rate and the signal (given as a vector) is returned. The sample rate is used to calculate the length (time in seconds) of the song. The signal is sampled at each instance in time. The number of samples taken (variable n in the code) is given by (sample rate)*(length of song). Note that this number is odd, which changes how the frequencies are vectorized. The domain (or L) is the length of the song, since there are n sampled points between the beginning and end of the song. The frequency domain is then initialized. When using FFT, it will automatically scale all values by 2π, so the frequency domain (given by k) is scaled to $\frac{2\pi}{L}$. With an odd number for n, the frequency domain must also be adjusted to accommodate the

frequency domain initialization. A comparison of k between even and odd values for n is given in the code in Appendix B. Due to the ordering when using fft( ), fftshift( ) must be used when plotting these signals.

To explore the Gabor transform and spectrograms, different parameters of the Gabor transform were studied. In particular, three different widths of the Gabor transform (the window function used being a Gaussian) and five different $\Delta\tau$ (see Section II) values were studied. All combinations of these widths and $\Delta\tau$ were studied using the spectrogram. Thus, a for loops over these widths and $\Delta\tau$ looped over all potential combinations. For a particular $\Delta\tau$ and width combination, the Gabor transform was calculated by multiplying the window function (the Gaussian centered at each τ) by the signal, and taking the FFT to determine the frequency content at that particular τ. The Gabor transform is then taken at the next τ, continuing until the entire song has be studied (all τ values studied). The transformed signal at each τ is recorded, which are used to plot the spectrogram. Spectrograms for each combination were plotted using pcolor( ) and other plotting functions, and the discussion of these is given in Section IV.

Part II

In this part, the goal is to reproduce the music score of the music and look into how overtones affect time-frequency analysis. The algorithm is largely synonymous with the first part, but there is some extra code that deals with filtering the overtones. In addition, the spectrograms in this section take some time to generate, so the spectrogram commands are separated from the main body of code. Another feature of this algorithm is that it works for both the recorder and piano audio files, which just need to be switched out in the first line of code.

In the first portion of this algorithm, the audio file is read by Matlab using the audioread( ) command. The other initialized variables are determined similarly to that in Part I, since audioread( ) returns the signal given as a vector and the sampling rate. The main difference is that since n is even, k is initialized according to n being even. In addition, since the resulting frequencies are presented in Hertz in the problem statement, the frequencies are scaled by $\frac{1}{L}$. The parameters for Gabor were determined through empirical tests. There is also an extra parameter, a_overtone, which is the width of the Gaussian filter applied on the transformed data to remove overtones.

To build the unfiltered overtone spectrogram, the process is the same as in Part I. To determine the music score, however, the frequency content must be analyzed further. At each τ, the maximum magnitude will correspond to the frequency of the music note played (if there is a note being played), since the music note's frequency will be the most dominant frequency. When no music is played, the maximum magnitude is likely going to be noise (which will show clearly as black on the spectrogram). Although the frequencies are very distinct, as shown in Figure 2C and D, the frequency value determined from the maximum magnitude is used as a sanity check to compare with the frequencies given in the problem statement.

To build the filtered overtone spectrogram, the same process is taken as above, however, after the Gabor transform is taken, another Gaussian filter is applied on the frequencies (Fourier filtering). The filter is applied around the maximum frequency signal, to remove frequencies outside of this frequency. These filtered frequencies are plotted in the spectrogram.

Section IV – Experimental Results

Part I – Exploration of Time-Frequency Analysis Parameters and Techniques

In this part of the problem, the main goal was to explore time-frequency techniques. To get the best idea of how different parameters might change the resulting spectrogram, I constructed a "matrix" of spectrograms. This is shown in Figure 1 below, where the a is a measure of width (inversely proportional to width) of the Gabor transform window and $\Delta\tau$ is given in Section II. Note the units indicated in the caption. A Gaussian window function was used for these Gabor transforms.

As the a increases, the width of the Gabor transform is decreases, meaning that there is more information about time (refer to Section II), but less about frequency. When a is small (large width), there is much more high frequency content compared to when width is small. In addition, when a is small, the frequencies look as if they "bleed" together more, indicating that there is more uncertainty as to when exactly that frequency occurs in time. At larger values of a, there is less of this phenomenon since there is more information about time.

As $\Delta\tau$ increases, oversampling is decreased. However, extremely small values for $\Delta\tau$ do not increase the amount of information provided as compared to larger $\Delta\tau$ values. However, this is limited at very large $\Delta\tau$ where there is very little information captured because so few Gabor transforms are taken. In the cases of undersampling (such as a = 3 and $\Delta\tau = 2$), information is actually lost. This can be seen as the sharp discontinuities of the frequencies (such as at the 4 second range).

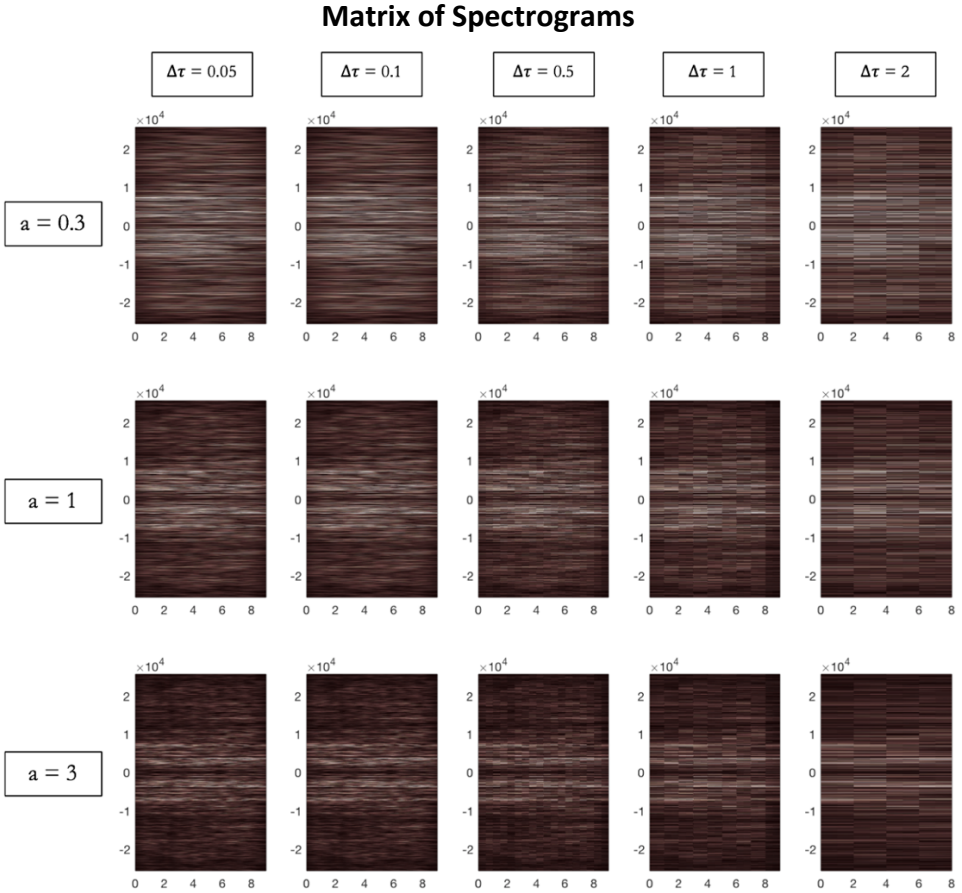

Figure 1: Matrix of spectrograms produced for Handel's song. Each spectrogram is indexed and characterized by the a (width) parameter and the $\Delta\tau$. The x-axis for each spectrogram is time (length of song, measured in seconds) while the y-axis is the frequency (rad/sec)

Part II – Time-Frequency Analysis in Music

In the first problem of this part, the music score is reproduced. One clear way to see the music score without understanding how to write music is by reproducing the spectrograms with the appropriate notes filled in. Produced below are 4 spectrograms, two without filtering overtones and two with filtered overtones for each of the two instruments. The frequencies were negated, and then matched to the frequencies given in the problem statement to reproduce the score. The reason for the negation is a result of how the max( ) function was used in the code. The spectrogram for the piano rendition indicates that notes such as the 4th and 7th E, the 5th D, and the last C were played longer than the other notes. The last C was likely played three or four times longer, while the other notes were likely played twice as long. Similarly, in the recorder recording, the last G looks as if it is also played three or four times longer than the other notes.

The main difference between the two instruments is that they play different notes, yet have the same tune of Mary had a Little Lamb. In addition, looking at the two spectrograms of the unfiltered overtones, the two instruments also have very different character. For the recorder, barely any timbre exists in the instrument since there are no visible overtones present. With the piano, the overtones exist at different multiples of the base frequency.

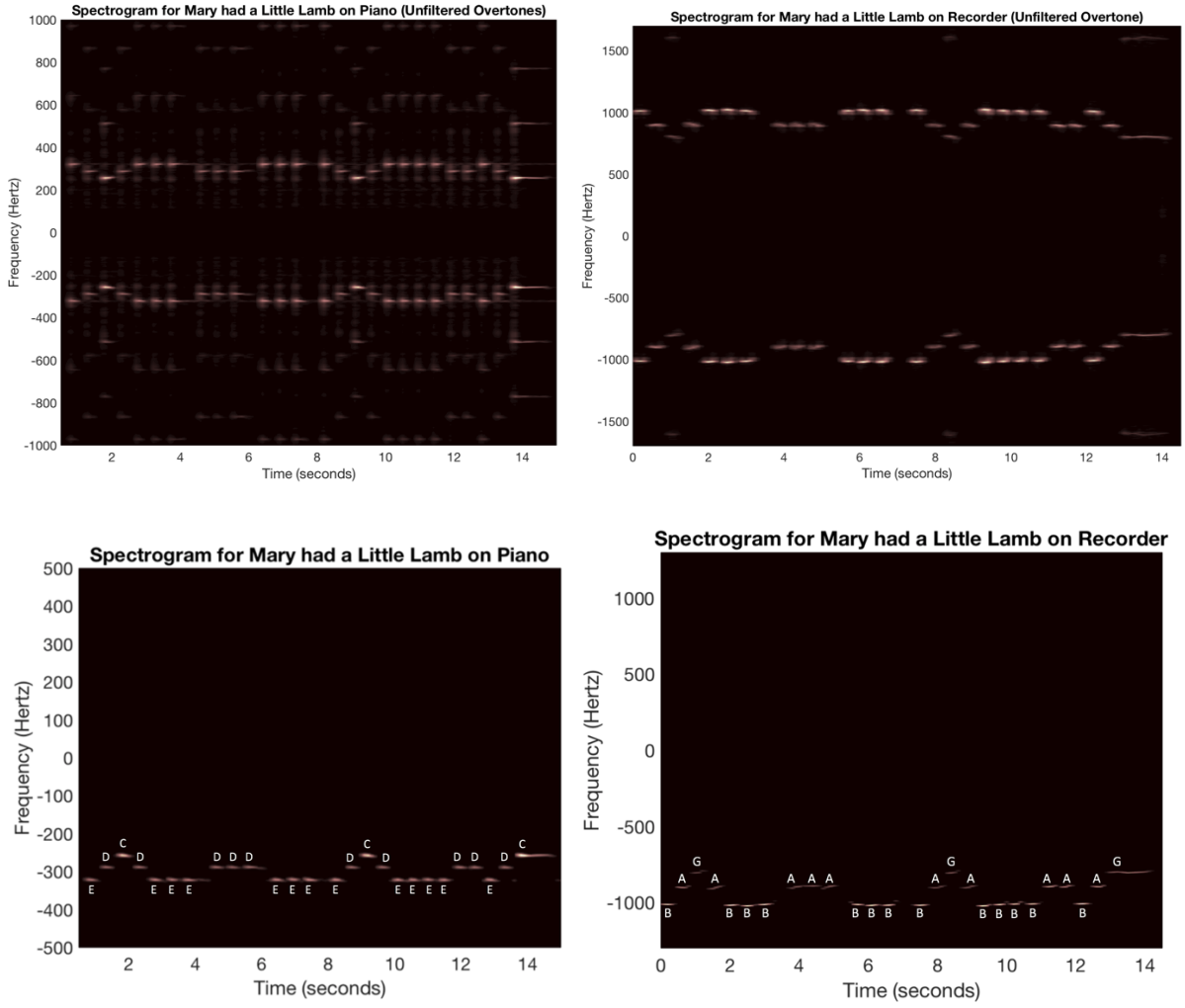

Figure 2: Spectrograms of "Mary had a Little Lamb", 2A: spectrogram of unfiltered overtones on piano, 2B:

spectrogram of unfiltered overtones on recorder, 2C: spectrogram of filtered overtones on piano, 2D: spectrogram of filtered overtones on recorder

Section V – Summary and Conclusions

Time-frequency analysis tools can be used to study signals that occur across an expanse of time. The Gabor transform in particular is one way to best digest a signal and decompose it into different sections to be analyzed separately (by Fourier analysis). At each instance in time, frequency information occurring at that instance in time can be captured. One of the limitations of this technique is that not all time and frequency information can be effectively captured (reducing $\Delta\tau$, or the spacing between taking Gabor transforms, can improve resolution but takes Matlab a long time to run). However, in this analysis, extreme values for $\Delta\tau$ do not need to be taken to present very similar spectrograms, as shown in the first part of the assignment, so there is an effective visual limit to how much resolution can be improved. In the second part of the assignment, however, the frequency characteristics of the signal is very distinct, making the assignment for $\Delta\tau$ to be more lenient. In fact, this frequency characteristic is so distinct (only one note is played at a time), that the entire music score can be accurately reproduced, showing how accurate Gabor transforms can be even with the uncertainty in time/frequency resolutions.

Appendix A

- **fft( ) –** takes Fast Fourier Transform in one-dimensional space
    - Used in the first and second parts of the problem to create the Gabor transform to determine the frequency character of the music signals
- **fftshift( ) –** shifts the values given by the fft( ) function so that they match those regularly used (mostly for plotting purposes)
    - fft( ) will shift the frequencies (in vector form) so they start from 0, increasing to the most positive value, and then start from the most negative value and increase to 0
    - fftshift( ) will be used so the frequencies start with the smallest (most negative to most positive, which is how plots are generally constructed)
- **subplot(m, n, x) –** plotting command, plots the visualization in different sections of the figure window
    - Used in the spectrogram matrix (Section III: Part I), where m*n gives the number of equally sized spaces set aside in the figure window, and x is the position of a particular space
- **pcolor(t, f, S) –** plotting command, used to build a spectrogram and display matrix data as an array of colored cells
    - Input arguments used for the purposes of this assignment:
        - t is a vector of times where a Gabor transform is taken at each time value
        - f is a vector of all frequencies considered by Gabor transform
        - S is the magnitude of frequencies created after taking the Gabor transform at a particular instance in time, essentially, take Gabor transforms and lay them vertically and consecutively in time
- **shading interp –** plotting command, used so pcolor displays colors using interpolation between neighboring data points
- **set( ) –** plotting command, used to set limits on the axes
- **colormap( ) –** plotting command, used to set the colors and color range of the resulting spectrogram from pcolor( ), where the color represents the magnitude of the information plotted
    - Used colormap(pink) since it was easiest to see high frequency information with these colors (on the particular laptop used to generate the data)
- **drawnow –** plotting command, used to
- **[y, Fs] = audioread( ) –** used to read in the audio file from local desktop, which returns the sampling rate (Fs) and the sampled data (y)
    - Used to read in the Mary had a Little Lamb audio files into Matlab
- **[M, I] = max(U(:)) –** used to find the maximum value and the index of the maximum value
    - For each particular instance in time, the maximum value of the Gabor transform indicates which frequency is most present at this instance in time
    - I, or index, helps locate what exact frequency this is