Gábor Transforms

University of Washington
AMATH 582 Homework 2
Winter Quarter 2019

David Long
dmlong888@live.com

Abstract

The Fast Fourier Transform, Gábor sampling, frequency filtering, and Multiple Resolution Analysis are explored in this decomposition of the timeless classic, “Mary Had a Little Lamb.” Through the use of spectrograms, two different versions of the tune will be parsed and notated.

I. Introduction and Overview

One would be hard-pressed to meet an individual who does not enjoy listening to music at least occasionally. Music is a tremendously impactful medium through which human beings are able to communicate profound emotions and feelings to fellows. It is a form of communication that transcends language and epochs. The symphonies of Beethoven can take an engaged listener on an epic journey. John Coltrane's sheets of sound look boldly into the abyss of the human experience and reflect what so many of us are reluctant to gaze upon. The relentless funk of James Brown is sure to induce rhythmic head-bobbing at the least, spasmodic dancing at the extreme. As such, music is regarded as a high form of art and esteemed careers are built upon its study and performance. Despite this classification, the physics of music belong to the mathematicians. The fact that this emotional medium can be so precisely quantized is something of a paradox. A successful musician need know nothing about the underlying physics to convey his message. Similarly, scientists who advance the mathematical understanding of acoustics may be entirely unable to carry a tune, let alone shred a guitar with ruthless precision.

This paper will explore some of the mathematical considerations in digital music decomposition. A short audio sample of Handel's "Messiah" will be used to explore the time-frequency relationship and to develop computational tools that can be applied to other samples. These tools will be used to decompose two different versions of the seminal classic "Mary Had a Little Lamb." The resultant time-frequency distribution will be used to construct a musical score of the tune.

II. Theoretical Background

When the name Pythagoras (c. 570 – c. 495 BC) is mentioned, both mathematicians and civilians alike will likely think of the Pythagorean Theorem, which concerns the relationship between the sides of a right triangle. Considering his influence in the development of the diatonic scale, upon which all of Western music is based, his reputation may be deserving, but misplaced and understated.

Modern music is based on the concepts of consonance and dissonance. Certain groups of notes sound consonant and resonate with one another, sounding resolved and at rest, while others will pull and grate at each other. The ear begs for resolution. It is said that Pythagoras was prompted to investigate this phenomenon when he walked past a blacksmith's shop and observed that anvils of different sizes produced different pitches[3]. Using the lyre, he was able to explore the frequencies produced by strings of various lengths and to quantify the levels of consonance. It was found that the highest level of consonance occurs when pitches have frequencies that are integer multiples of each other. The interval that sounds the most consonant is the octave, which occurs when the multiple is a power of two. The second-most consonant interval occurs at a multiple of 3, which is known as a perfect fifth. The most dissonant interval is considered to be

the diminished fifth, which occurs at a half-step below the perfect fifth. Though the frequency spectrum is continuous, the human ear has difficulty in differentiating between frequencies that are close together. The twelve-note octave was soon discretized and the available levels of consonance and dissonance were thus established. This discretization is the foundation of all Western music.

Perhaps the reason that our ear perceives some intervals as more consonant than others is due to the physical phenomenon known as overtones. When a fundamental frequency such as 261.63 Hz is struck on a lyre, the vibrations create nodes along the length of the string that produce additional frequencies beyond the fundamental. The first two overtones above a fundamental are the octave and a perfect fifth above that. We are used to hearing those overtones above every note that is struck. They sound like they belong together because they occur together naturally. Upon considering the possibility of striking more than one note at a time and the overtone series that are generated by those combinations, it becomes apparent that a vast array of levels of consonance and dissonance are possible, as evidenced by the tremendous diversity in the catalog of Western music that is built upon the same twelve-note division of the octave.

The human ear receives sound as analog waves. Recording mediums such as vinyl and magnetic tapes preserve the full scope of those waves. Through psychoacoustic experimentation, researchers have determined that there is but a small range of wavelengths that can be perceived, roughly 20 – 20,000 Hz. Furthermore, that range is quantized into 24 frequency bands that are distinguishable to the ear. The dominant presence of a tone in one of the bands has the tendency to mask other tones that may be present in the same band[2]. Digital music exploits these limitationa by attempting to retain the minimum amount of information essential to the analog wave structure.

The identification of frequencies in a signal is made possible through the Fast Fourier Transform (FFT), which allows data to be represented in the frequency domain by integrating the product of the signal $f(t)$ and a kernel $g(t) = e^{-ikt}$ over the time domain.

$$\Phi\{f(t)\} = F(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-ikt} dt \qquad \text{(Eqn. 1)}$$

While this is germane to our objective, the process of transforming the data to the frequency domain sacrifices the information that is gathered over time. A modification of the FFT was developed by Hungarian physicist Gábor Dénes to address this deficiency. The method takes several small samples of the data across time and localizes the frequency content with successive FFTs, a process which captures both time and frequency data and gives rise to its being known as the short-time Fourier transform. The upgrade is accomplished by a modification of the kernel, which creates a function of both time and frequency.

$$\Gamma\{f(t)\} = G_g(t,k) = \int_{-\infty}^{\infty} f(\tau)\, g(\tau - t)\, e^{-ikt}\, d\tau \qquad \text{(Eqn. 2)}$$

The factor of $g(\tau - t)$ creates a sampling window where localized frequencies can be measured. The function is chosen to zero out frequencies that are far from the current value of *t*. As *t* is slid from the beginning of the signal to the end, the integral over all time τ gives information about the current distribution of frequencies. Typical functions that may be used for *g* include Gaussians and Mexican Hats.

Figure 1 illustrates some typical Gábor sampling windows and their corresponding frequency distributions. The diagrams on the left are of a Gaussian filter in the form of

$$g(\tau - t) = \exp\left[-a(\tau - t)^2\right], \qquad \text{(Eqn. 3)}$$

where *a* controls the spread around the center of the sampling window *t*. The diagrams on the right are of a Mexican Hat filter. The Mexican Hat wavelet filter is given by

$$g(\tau - t) = \left[1 - a(\tau - t)^2\right] \bullet \exp\left[-\frac{a(\tau - t)^2}{2}\right] \qquad \text{(Eqn. 4)}$$

The largest drawback to the short-time Fourier transform is that when the window is too small, low frequencies of wide wavelength may be missed in the signal. The process of decomposing a signal can then become a balancing act between appropriate resolutions in both time and frequency. Too much resolution in one can cause too little in the other. While the method is surely an improvement over the FFT, the compromises that ensue due to the fixed window width eventually led to the formulation of wavelets as a method of analyzing the time-frequency content of a signal.

The premise of wavelets is to allow the width of the sampling window to vary. The *mother wavelet* is a two-parameter function of time that satisfies

$$\psi_{a,t}(\tau) = a^{-1/2}\psi\left(\frac{\tau - t}{a}\right) \qquad \text{(Eqn. 5)}$$

ψ is essentially a Gábor sampling window with a scaling factor. The window is first chosen so as to be very wide and capture the entire frequency spectrum. Progressively, the next window is chosen so as to be orthogonal to the last. These progressive decompositions form a basis for the signal and the process of constructing it is known as Multiple Resolution Analysis. The signal $f(t)$ is then expressed by the sum

$$f(t) = \sum_{a,t=-\infty}^{\infty} c_{a,t}\psi_{a,t}(\tau) \qquad \text{(Eqn. 6)}$$

where $c_{a,t} = \left(f, \psi_{a,t}\right)$, the inner product of f and ψ. While the expansion is exact in the limit, just a few values of *a* are often enough to adequately approximate almost all of the signal. For conditions that need to be satisfied to ensure that the wavelet function produces a basis decomposition, see [1]. For an investigation of Haar wavelets, see Appendix A.

III. Algorithm Implementation and Development

Before deconstructing "Mary Had a Little Lamb," I got some practice in making spectrograms by exploring a sample of Handel's "Messiah." The first steps were to import the file and calculate the computational domains. MATLAB provides a signal vector and a sample rate so the length of the time domain $(0, L)$ can be calculated in seconds by dividing the length of the signal vector by the sample rate. To account for the assumed periodicity of the Fast Fourier Transform, only the first *n* points of `linspace(0, L, n+1)` were used for the time domain. The Fourier modes were shifted and scaled by `(2*pi/L)` to account for the FFT algorithm. For visualization on the Hertz scale, the Fourier domain was divided by `2*pi` to give units of waves per second. At this point, I also set up a vector of filter widths and a vector to slide the Gábor windows across the time domain.

The sampling window was defined as a function of time. Eqn. 3 and Eqn. 4 were coded and a double-loop moved the window along the time axis by changing the value of *t*. I explored different widths by changing the value of *a*. At each value of *t*, the evaluated sampling window was multiplied to the signal vector. The resultant vector was given an FFT and shifted, then its absolute value was appended to a matrix that stored the frequency data. At the end of the time-slide, the matrix was visualized as a spectrogram.

What followed next was really a lot of trial and error. A variety of Gábor window types and widths were used to plot the frequency spectrum versus time to explore their impacts on the resolution of the decomposition. I began with Gaussian filters in the form of Eqn. 3. As the value of *a* is increased, so is the resolution in time, but it comes at the expense of resolution in frequency. It quickly became apparent that the appropriate balance between the two would require some fine-tuning.

Figure 2 shows a matrix of Gaussian-filtered spectrograms. One can see that a low number of samples such as ten creates a spectrogram that looks very discrete, whereas over a high number of samples, the distribution smooths out over time, which seems to be a more realistic depiction of a continuous variable. Furthermore, for a fixed window width, increasing the number of samples reveals more of the frequencies that are present in the signal. The increased resolution

in both time and frequency does not contradict the FFT conundrum, but is consistent with basic statistical sampling theories. The figure also reveals that a wide time window gives a frequency resolution that is too dense to parse the fundamentals. The individual voices of the choir are indiscernible. Increasing the time resolution by compressing the sampling window reveals the choral range. With $a = 64$ and 100 samples, one can see both the individual voices and the richness of harmony and overtones that are present in the signal.

With regards to the filter width and number of samples, **Figure 3** reveals behavior similar to the Gaussian filter. As the Mexican Hat filter seems to be looking ahead and behind the current sample, I expected it to have better resolution in time than the Gaussian filter, but that was seemingly not the case. The Gaussian filter appears to give a clearer indication of where one note begins and the next ends. Contrary to my expectation, a comparison of the left columns in Figure 2 and Figure 3 reveals that the Mexican Hat wavelet is denser in frequency than the Gaussian.

With some familiarity of the behavior of the Gábor windows, I next turned my attention to the decomposition of "Mary Had a Little Lamb." I re-used all of the code from the Handel explorations to process both the piano and recorder versions.

The frequency plot of the piano version is rich with harmonic overtones. The density of frequencies between about 2,000 and 4,000 can be seen in **Figure 4**. The spectrogram in **Figure 5** shows intense frequencies near the fundamentals and at integer multiples on the Hertz scale.

Figure 6 shows that the intensity of overtones on the recorder is low compared to the piano. The spectrogram in **Figure 7** reveals the same step-pattern and lengths of notes as Figure 5. The overtones are of low intensity and the fundamentals fail to radiate for the recorder as they do for the piano.

Zooming in on the range of the fundamental frequencies gives a clearer indication of which notes on the piano are being played (**Figure 8)**. The notes that the frequency peaks in the spectrogram are closest to are at 329.63, 293.66, and 261.63 Hz (E, D, and C). The piano may be a little out of tune, but as previously noted, small differences from the target are difficult to discern audibly. For the recorder (**Figure 9**), the notes that the frequency peaks in the spectrogram are closest to are at 987.77, 880.00, and 783.99 Hz (B, A, and G).

The overtone series can easily be eliminated from the signal with a band-pass filter. In the frequency domain, lower and upper limits were set at 1,400 and 2,200 respectively to bound the notes actually being struck on the piano. The filter was initialized to zeros, then a for-loop traversed the spectrum and set the value to one any time there was a frequency in bounds. The signal was Fourier transformed, multiplied by the filter and inverse transformed before the Gábor sampling to produce **Figure 10**.

```
for j=1:length(filter)
    if ((ks(j) >= klower) && (ks(j) <= kupper))
        filter(j) = 1;
    end
end

Spf = ifft(ifftshift(filter.*fftshift(fft(Sp))));
```

I used the open-source tablature program TuxGuitar[4] to chart the songs in standard notation **(Figure 11** and **Figure 12)**.

IV. Computational Results

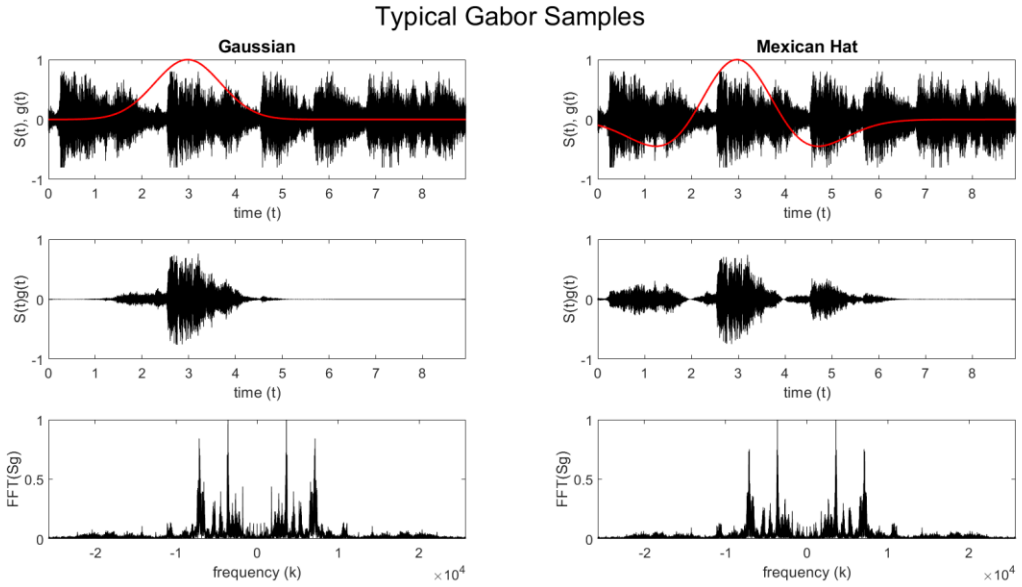

Figure 1: The red lines outline sampling filters, which are multiplied to the signal to produce the localized audio samples beneath. The frequency plots at the bottom are for the given samples.

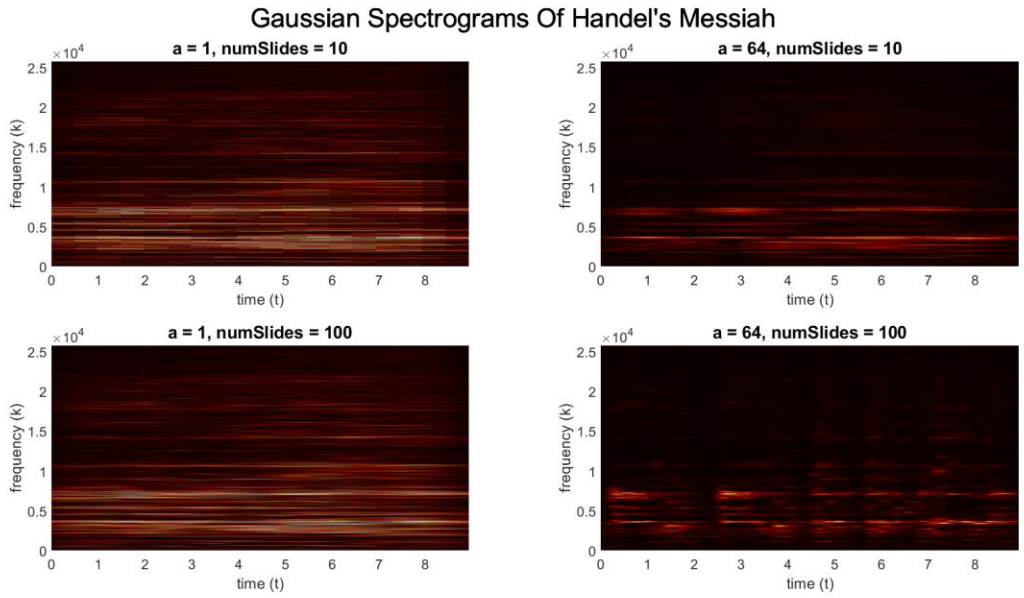

Figure 2: Within columns, it can be seen that increasing the number of samples in the time slide yields more information about both time and frequency. Within rows, compressing the filter width reduces the richness of the captured frequencies, but gives better time resolution.

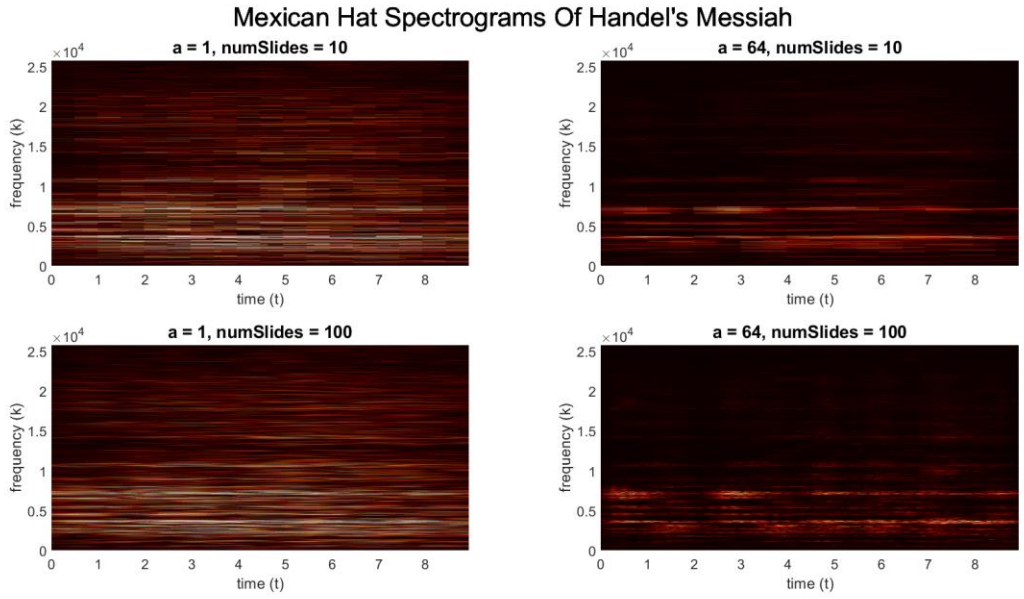

Figure 3: The Mexican Hat sampling filter does not seem to delineate the beginnings and endings of notes as well as the Gaussian filter of Figure 2, but a comparison of the plots at lower and higher numbers of samples reveals that it captures frequencies well.

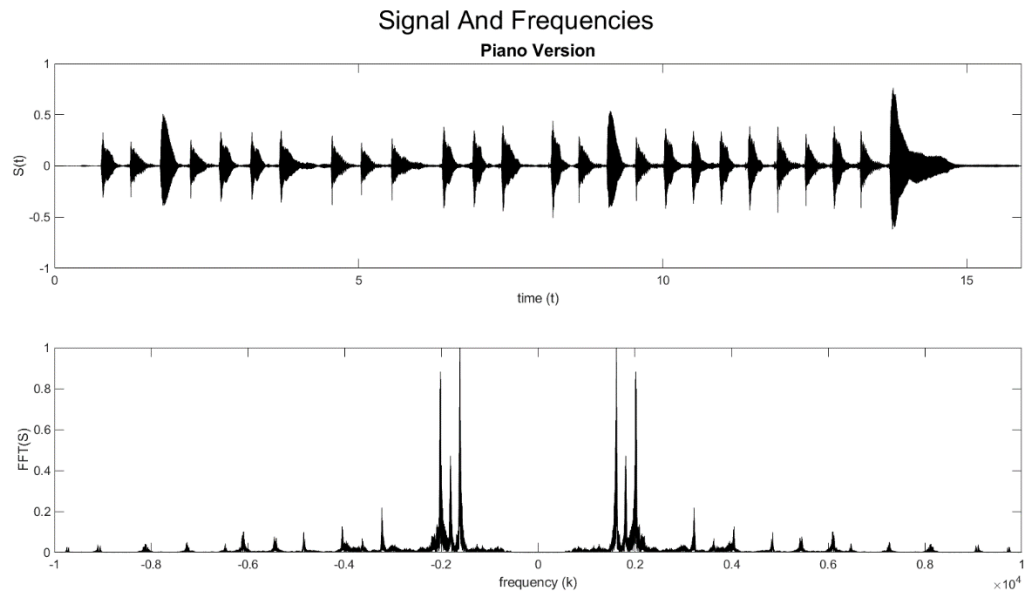

Figure 4: From the signal plot, it is observed that each note peaks in volume at the front of the attack and decays. The final note is the loudest and longest. The high-intensity frequencies are the fundamentals. Overtone activity is present at integer multiples of the fundamentals.

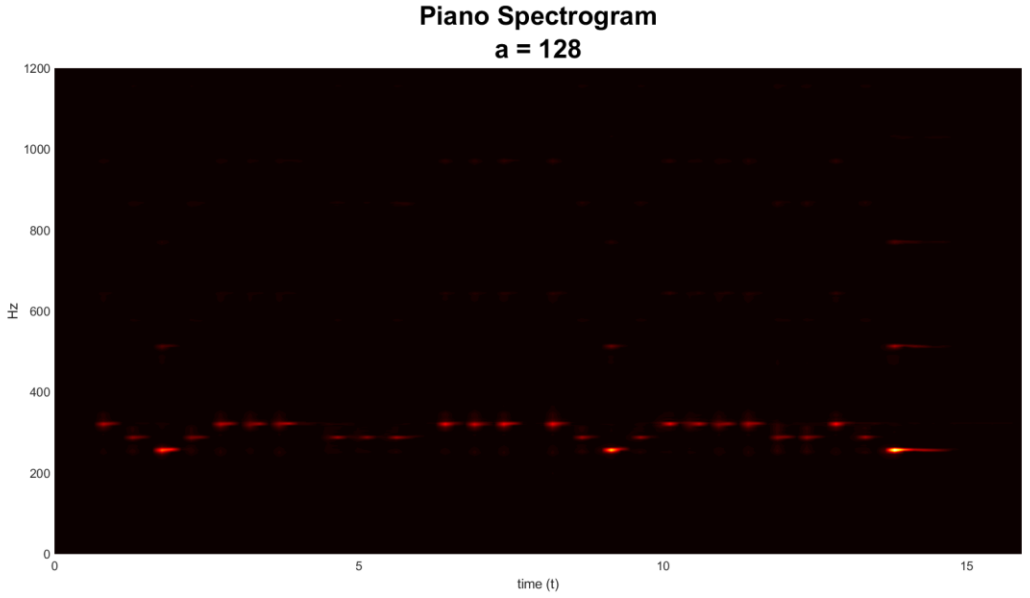

Figure 5: The fundamentals are near 300 Hz. Overtones can be seen stacked above them in integer multiples. The fundamentals themselves seem to stimulate adjacent frequency activity. Increasing the fundamental signal amplitude increases the intensity of the overtones.

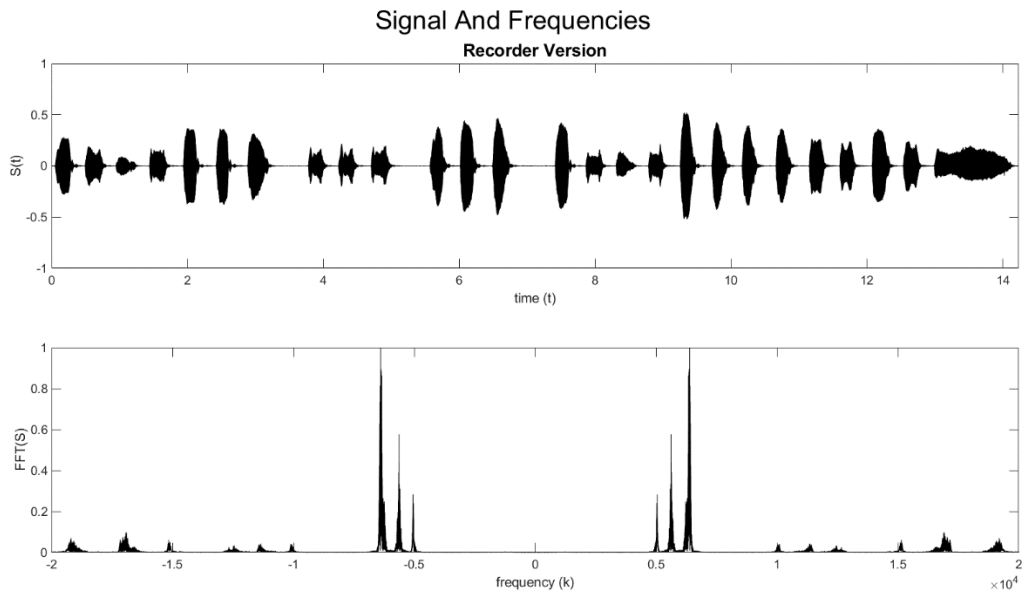

Figure 6: The signal amplitude for the recorder has less attack on the front end compared to the piano (Figure 4) and has periods of absolute silence between the notes. The overtone series is far less complex than that of the piano.

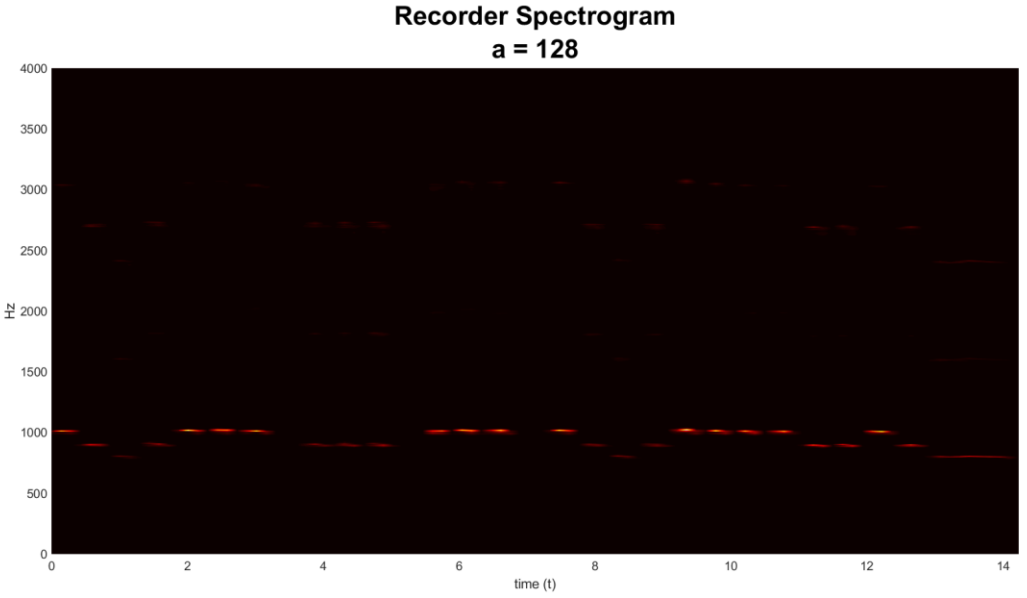

Figure 7: Compared to the piano (Figure 5), the fundamentals are narrow and isolated. Even the overtone series are low in intensity.

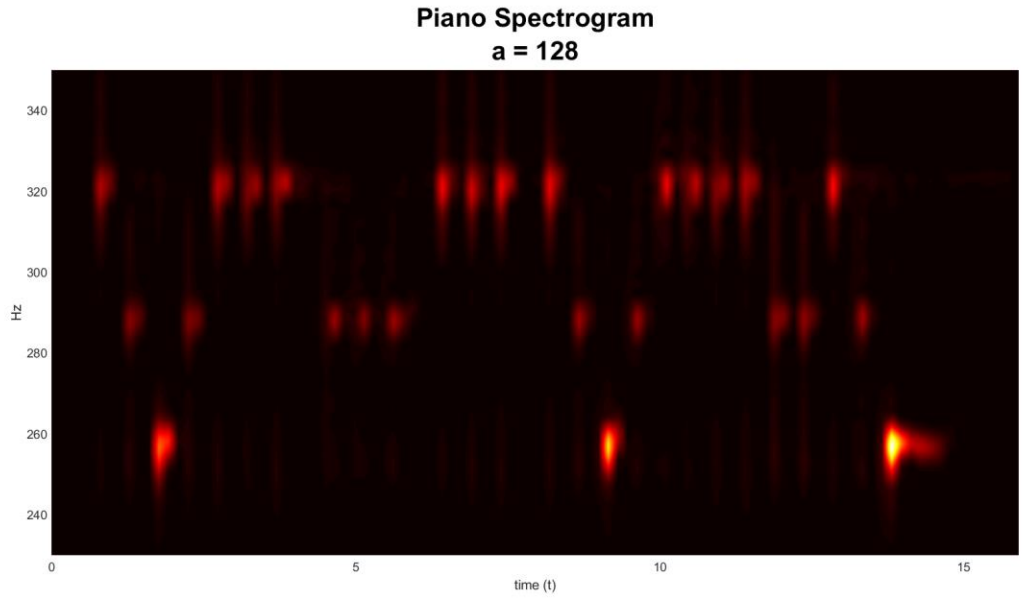

Figure 8: The notes that the frequency peaks in the spectrogram are closest to are at 329.63, 293.66, and 261.63 Hz (E, D, and C).

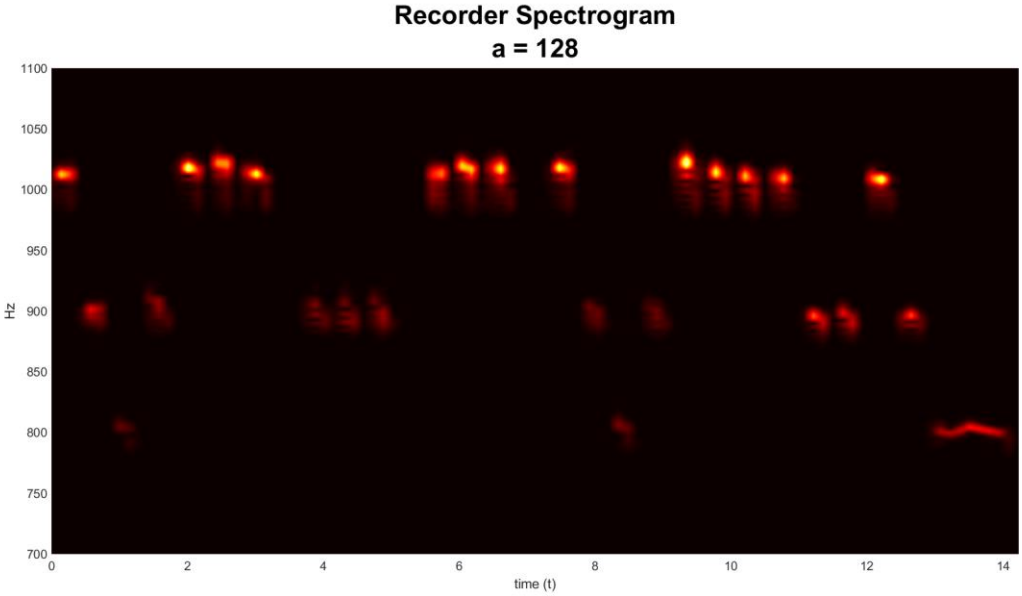

Figure 9: The notes that the frequency peaks in the spectrogram are closest to are at 987.77, 880.00, and 783.99 Hz (B, A, and G). Note the inconsistency of the pitches in the woodwind instrument compared to that of the stringed instrument.

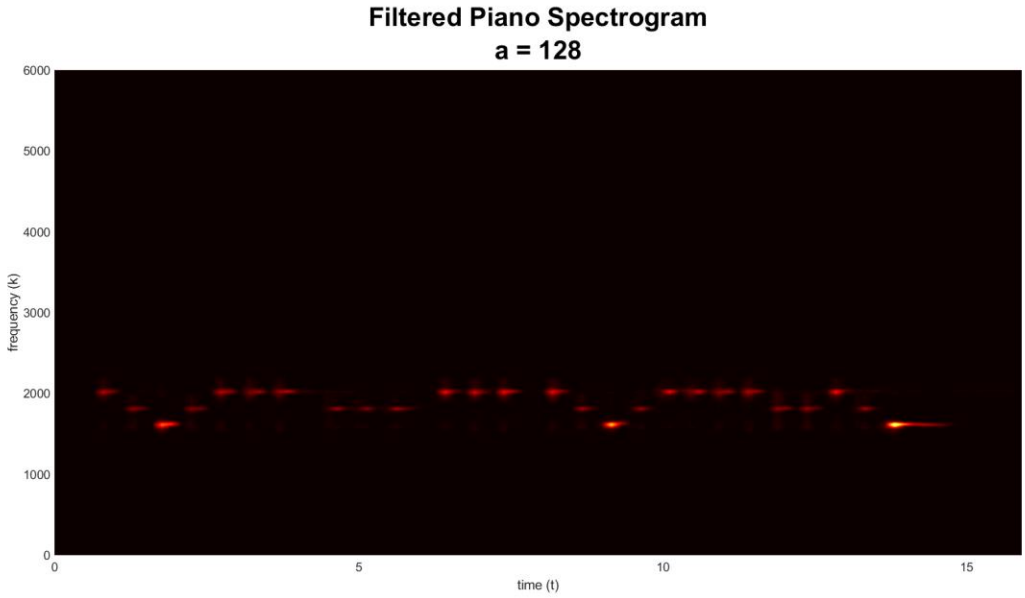

Figure 10: The piano version of "Lamb" with a band-pass filter applied to the frequency domain of 1,400 – 2,200. The filter was applied to the Fourier transform of the signal before Gábor sampling.

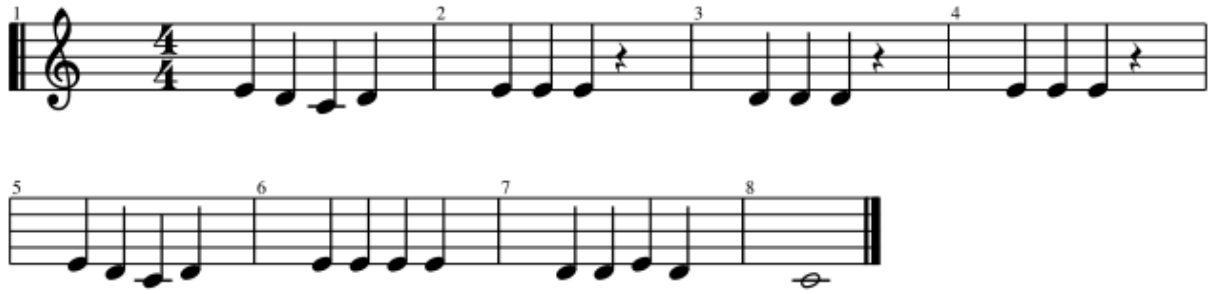

Figure 11: A transcription of the piano version of "Mary Had a Little Lamb," which was parsed from the spectrograms. The only three notes in the tune are, in descending order, E, D, and C.

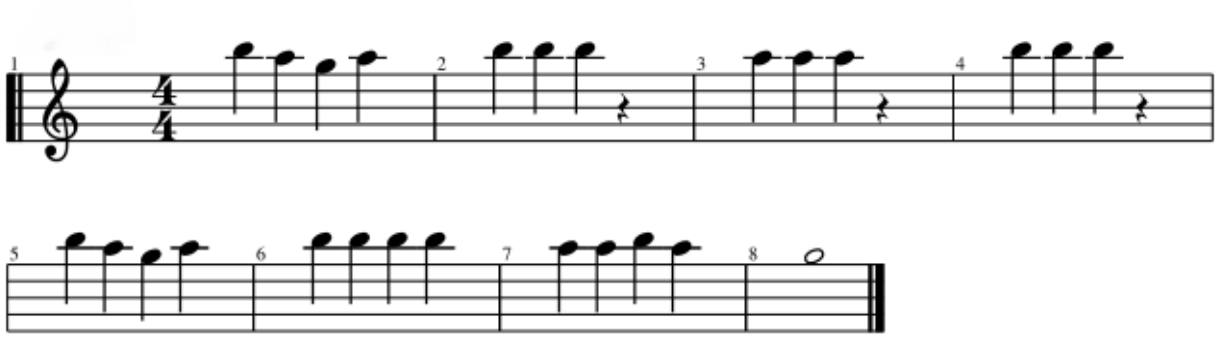

Figure 12: A transcription of the recorder version of "Mary Had a Little Lamb," which was parsed from the spectrograms. The only three notes in the tune are, in descending order, B, A, and G.

V. Summary and Conclusions

The Fast Fourier Transform was a revolutionary discovery, but in its pure form, it is best suited for applications with stationary frequencies. The work of Gábor makes the transform useful in analyzing how the frequency content of a signal changes over time and the Multiple Resolution Analysis makes the procedure even more precise. The computations are massive and resource-intensive, so it is only through recent advancements made in computer science that analyses such as these are even possible. While "Mary Had a Little Lamb" may not be everyone's favorite tune, the methods explored here are applicable to any digital audio file, so direct your receivers to the cosmos and begin decryption of the alien communications. If there is anyone out there shouting at us, it is probably in the language of physics.

A. Haar Wavelet Analysis

The Haar wavelet is defined as a piecewise step-function:

$$\psi(t) = \begin{cases} 1 & 0 \le t < 0.5 \\ -1 & 0.5 \le t < 1 \\ 0 & \text{otherwise} \end{cases} \qquad \text{(Eqn. A.1)}$$

It can be scaled, but must satisfy the two constraints $\int_{-\infty}^{\infty} \psi(t)\,dt = 0$ and $\int_{-\infty}^{\infty} |\psi(t)|^2\,dt = 1$. As an example of how the Haar wavelet can generate a basis for a signal using Multiple Resolution Analysis, a bit of the old Ludwig van was decomposed using MATLAB's built-in wavelet analysis tool. The first twenty seconds of "Symphony No. 5 in C-minor, Op. 67" is both dynamically and harmonically complex. **Figure A.1** shows the decomposition at five levels beyond the mother wavelet a_5. Several levels deep, it is observed that there are significant differences in the shapes of the d_i. It seems necessary to include all of the levels shown in the basis, perhaps even more. The spike that can be seen in d_1 is suspicious and possibly a computational artifact. In **Figure A.2**, the reconstructed signal is super-imposed over the original signal. It is mostly a very good approximation, though there are some slight discrepancies.

The piano version of "Mary Had a Little Lamb" was similarly put to the MRA with a five-level Haar sampling filter. **Figure A.3** shows very few differences between the d_i. **Figure A.4** shows that the super-imposition is almost exact. The last note of d_5 appears to deviate from the corresponding note in a_5, but rest of the subsets seem to all have the roughly the same shape. There is just one instrument playing in this piece, and it plays just three notes. The Ludwig van has far more frequency content, so it seems reasonable to expect an acceptable decomposition of it to incorporate more basis functions. A two-level MRA of "Lamb" is shown in **Figure A.5**. The super-imposition shown in **Figure A.6** makes one wonder if even that is overkill.

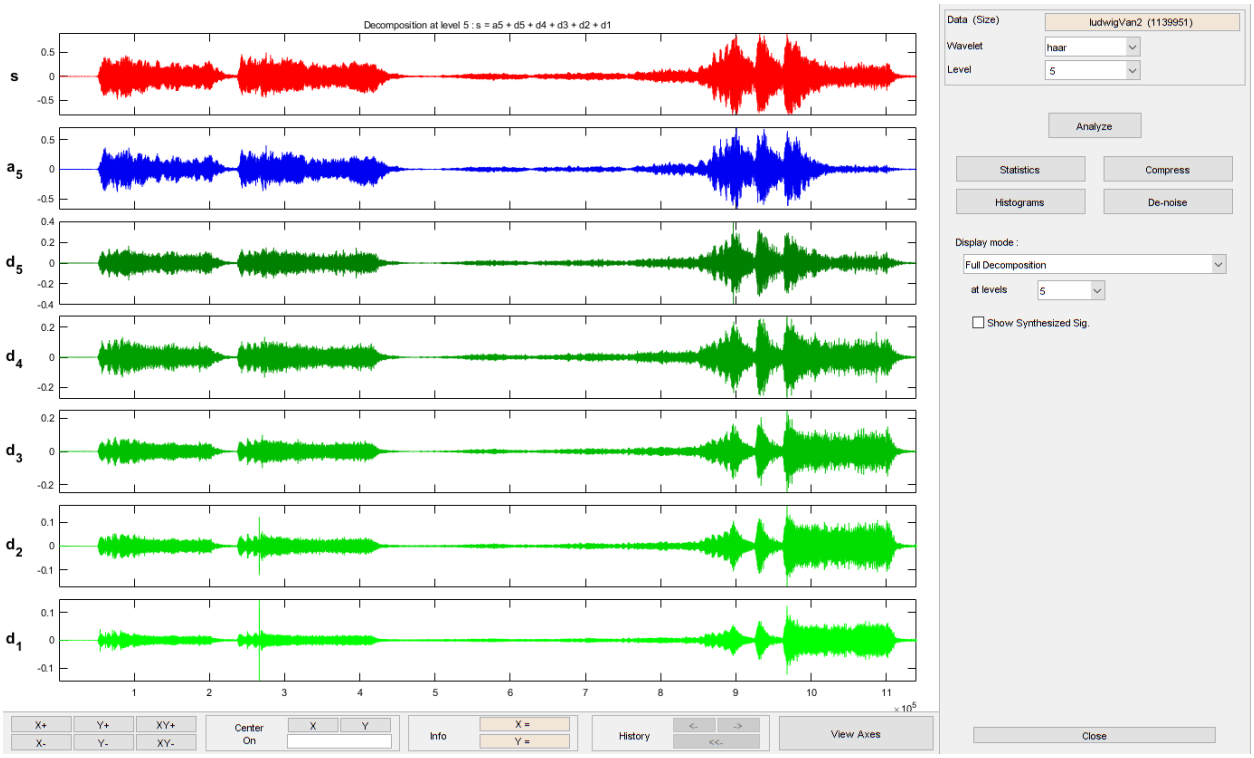

Figure A.1: A five-level MRA of Ludwig van Beethoven's "Symphony No. 5 in C-minor, Op. 67" using a Haar sampling window. All levels of the decomposition have observable differences from the others.

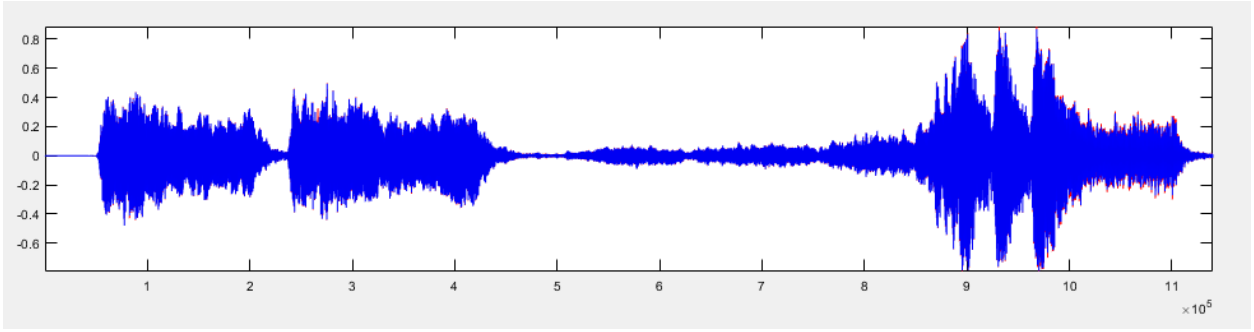

Figure A.2: The super-imposition of the reconstructed signal (blue) on the original signal (red) for "Symphony No. 5 in C-minor, Op. 67." It fits well, but there are some deficiencies. Finer resolution in the MRA would likely help the fit.

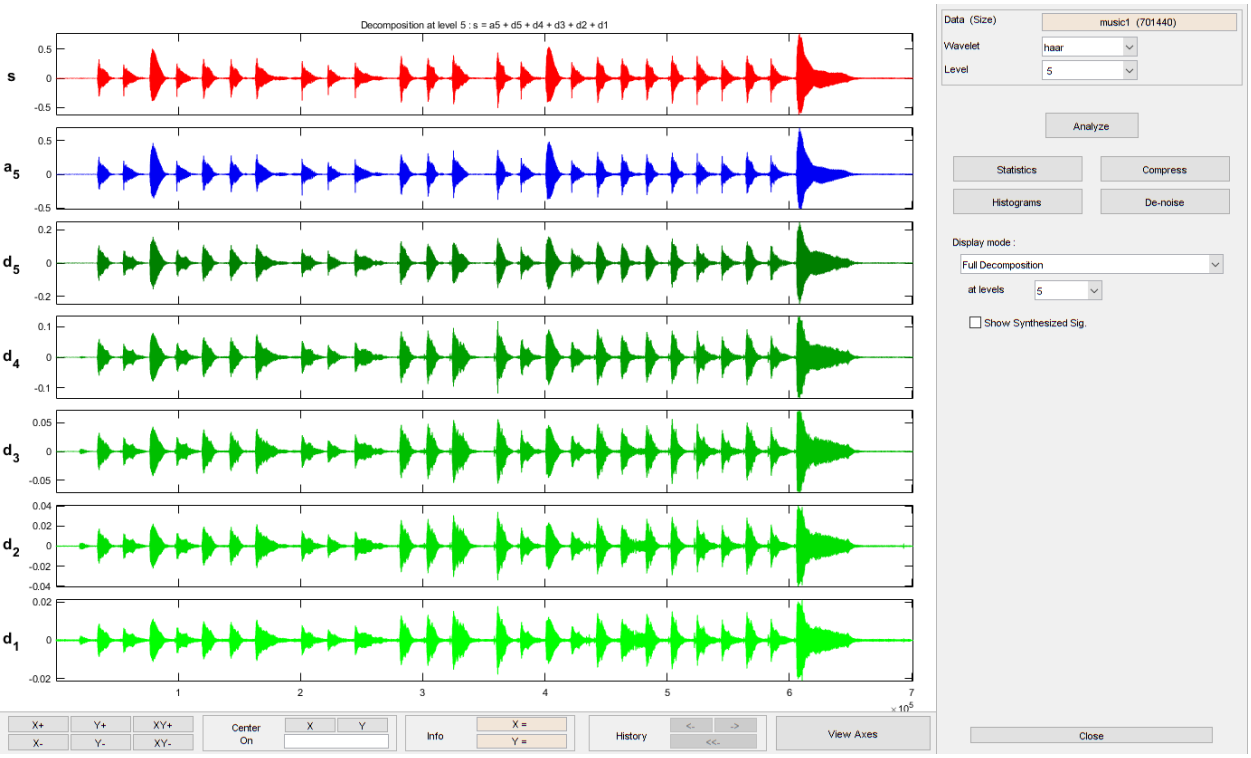

Figure A.3: A five-level MRA of the piano version of "Mary Had a Little Lamb" using a Haar sampling window. There appears to be more similarities in the d_i compared to the decomposition show in Figure A.1.

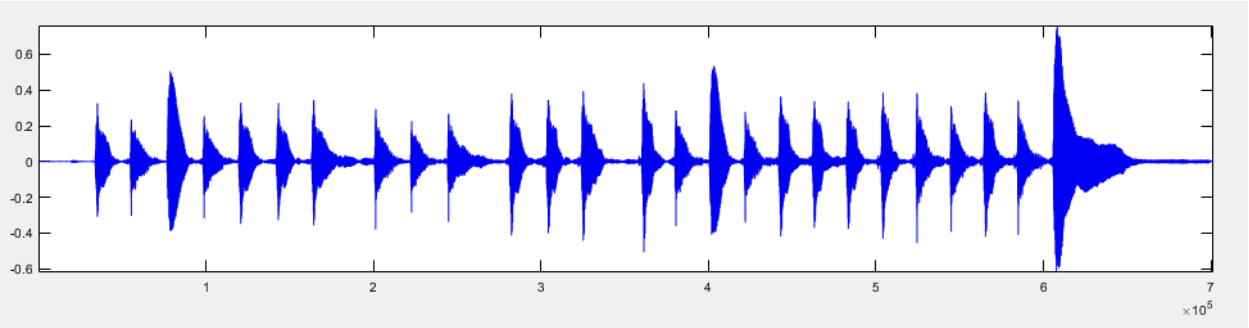

Figure A.4: The super-imposition of the reconstructed signal (blue) on the original signal (red) for "Mary Had a Little Lamb." The match is almost exact.

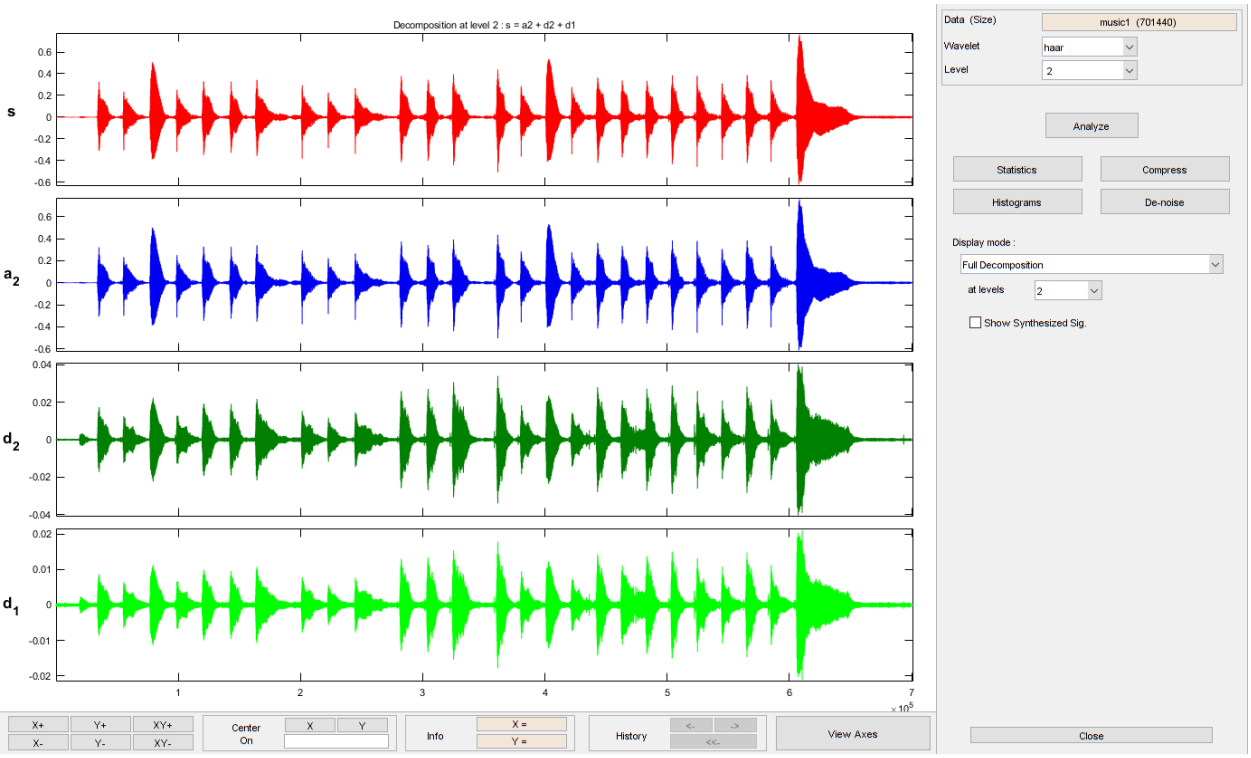

Figure A.5: The fit of the five-level MRA of "Mary Had a Little Lamb" was so good, a two-level basis was attempted. The d_i are similar to each other, but different from a_2.

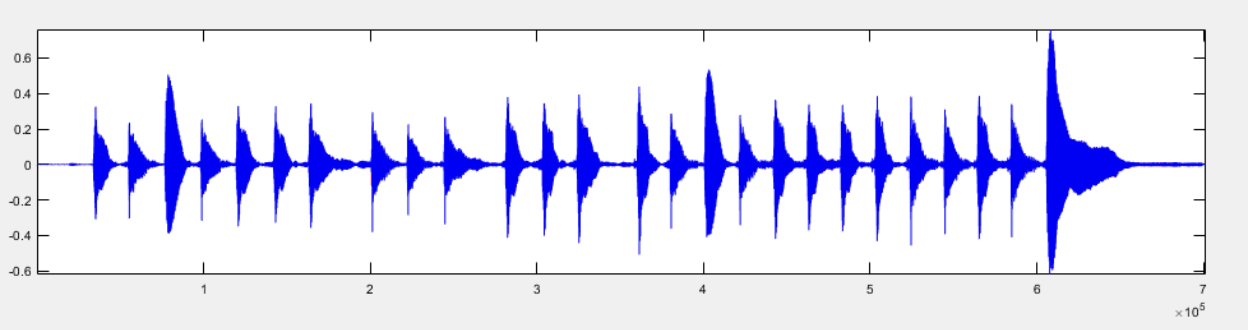

Figure A.6: The super-imposition of the reconstructed signal (blue) on the original signal (red) for "Mary Had a Little Lamb." The approximation appears to be every bit as good as the one generated by the five-level MRA in Figure A.4.

B. MATLAB Codes

```
% Homework 2 - Gabor Transforms
% David Long
% 1.20.19

clear all; close all; clc

% Load signal

load handel;
S = y.';




% Computational domain

L = length(S)/Fs;
n = length(S);
if (mod(n,2))==1
    S = S(1:n-1);
    n = n - 1;
end

t2 = linspace(0, L, n+1);
t = t2(1:n);

k = (2*pi/L)*[0:(n/2 - 1) (-n/2):-1];
ks = fftshift(k);

b = L/3;
width = [1, 64];




% Gabor windows

figure(1)
sgtitle('Typical Gabor Samples', 'Fontsize', 30)

% Guassian windows

g = exp(-width(1)*(t-b).^2);
Sg = g.*S;
Sgt = fft(Sg);

subplot(3, 2, 1)
plot(t, S, 'k'), hold on
plot(t, g, 'r', 'LineWidth', 2)
set(gca, 'Fontsize', 16, 'Xlim', [0 L], 'Ylim', [-1 1])
```

```
ylabel('S(t), g(t)'), xlabel('time (t)')
title('Gaussian', 'Fontsize', 20)

subplot(3, 2, 3)
plot(t, Sg, 'k')
set(gca, 'Fontsize', 16, 'Xlim', [0 L], 'Ylim', [-1 1])
ylabel('S(t)g(t)'), xlabel('time (t)')

subplot(3, 2, 5)
plot(ks, abs(fftshift(Sgt))/max(abs(Sgt)), 'k')
set(gca, 'Fontsize', 16, 'Xlim', [ks(1) ks(end)], 'Ylim', [0 1])
ylabel('FFT(Sg)'), xlabel('frequency (k)')




% Mexican hat windows

g = (1 - width(1)*(t - b).^2).*exp((-width(1)*(t - b).^2)/2);
Sg = g.*S;
Sgt = fft(Sg);

subplot(3, 2, 2)
plot(t, S, 'k'), hold on
plot(t, g, 'r', 'LineWidth', 2)
set(gca, 'Fontsize', 16, 'Xlim', [0 L], 'Ylim', [-1 1])
ylabel('S(t), g(t)'), xlabel('time (t)')
title('Mexican Hat', 'Fontsize', 20)

subplot(3, 2, 4)
plot(t, Sg, 'k')
set(gca, 'Fontsize', 16, 'Xlim', [0 L], 'Ylim', [-1 1])
ylabel('S(t)g(t)'), xlabel('time (t)')

subplot(3, 2, 6)
plot(ks, abs(fftshift(Sgt))/max(abs(Sgt)), 'k')
set(gca, 'Fontsize', 16, 'Xlim', [ks(1) ks(end)], 'Ylim', [0 1])
ylabel('FFT(Sg)'), xlabel('frequency (k)')




% Gaussian tslide loop

location = 1;
numSlides = [10, 100];

for jjj=1:length(numSlides)

    tslide = linspace(0, L, numSlides(jjj));
```

```
    % Dilation loop

    for jj=1:length(width)

        Sgt_spec = [];

        for j=1:length(tslide)

            g = exp(-width(jj)*(t - tslide(j)).^2);
            Sg = g.*S;
            Sgt = fft(Sg);
            Sgt_spec = [Sgt_spec; abs(fftshift(Sgt))];

        end




        % Spectrogram of time-frequency signal

        figure(2)
        sgtitle('Gaussian Spectrograms Of Handel''s Messiah', 'Fontsize', 30)

        Sgt_spec = Sgt_spec.'/max(max(Sgt_spec));

        subplot(2, 2, location)
        pcolor(tslide, ks, Sgt_spec), shading interp
        set(gca, 'Fontsize', 16, 'Xlim', [0 L], 'Ylim', [0 ks(end)])
        colormap(hot)
        xlabel('time (t)'), ylabel('frequency (k)')
        title(['a = ', num2str(width(jj)), ', numSlides = ',
num2str(numSlides(jjj))], 'Fontsize', 20)

        location = location + 1;

    end

end




% Mexican hat tslide loop

location = 1;

for jjj=1:length(numSlides)

    tslide = linspace(0, L, numSlides(jjj));
```

```
    % Dilation loop

    for jj=1:length(width)

        Sgt_spec = [];

        for j=1:length(tslide)

            g = (1 - width(jj)*(t - tslide(j)).^2).*exp((-width(jj)*(t -
tslide(j)).^2)/2);
            Sg = g.*S;
            Sgt = fft(Sg);
            Sgt_spec = [Sgt_spec; abs(fftshift(Sgt))];

        end




        % Spectrogram of time-frequency signal

        figure(3)
        sgtitle('Mexican Hat Spectrograms Of Handel''s Messiah', 'Fontsize',
30)

        Sgt_spec = Sgt_spec.'/max(max(Sgt_spec));

        subplot(2, 2, location)
        pcolor(tslide, ks, Sgt_spec), shading interp
        set(gca, 'Fontsize', 16, 'Xlim', [0 L], 'Ylim', [0 ks(end)])
        colormap(hot)
        xlabel('time (t)'), ylabel('frequency (k)')
        title(['a = ', num2str(width(jj)), ', numSlides = ',
num2str(numSlides(jjj))], 'Fontsize', 20)

        location = location + 1;

    end

end
```

```
% Homework 2 - Gabor Transforms
% David Long
% 1.20.19

clear all; close all; clc

% Load signal

[yp, Fsp] = audioread('music1.wav');
Sp = yp.';




% Computational domain

L = length(Sp)/Fsp;
n = length(Sp);

t2 = linspace(0, L, n+1);
t = t2(1:n);

k = (2*pi/L)*[0:(n/2 - 1) (-n/2):-1];
ks = fftshift(k);
kf = ks/(2*pi);

b = L/3;
width = [128];

numSlices = 100;
tslide = linspace(0, L, numSlices);




% Signal function and transform

St = fft(Sp);

figure(1)
sgtitle('Signal And Frequencies', 'Fontsize', 30)

subplot(2, 1, 1)
plot(t, Sp, 'k')
set(gca, 'Fontsize', 14, 'Xlim', [0 L], 'Ylim', [-1 1])
xlabel('time (t)'), ylabel('S(t)')
title('Piano Version', 'Fontsize', 20)

subplot(2, 1, 2)
plot(ks, abs(fftshift(St))/max(abs(St)), 'k')
set(gca, 'Fontsize', 14, 'Xlim', [-10000 10000], 'Ylim', [0 1])
xlabel('frequency (k)'), ylabel('FFT(S)')
```

```
% Dilation loop

for jj=1:length(width)

    Sgt_spec = [];

    % Translation in time
    for j=1:length(tslide)

        g = exp(-width(jj)*(t - tslide(j)).^2);
        Sg = g.*Sp;
        Sgt = fft(Sg);
        Sgt_spec = [Sgt_spec; abs(fftshift(Sgt))];

    end




    % Spectrogram of time-frequency signal

    figure(2)

    Sgt_spec = Sgt_spec.'/max(max(Sgt_spec));

    pcolor(tslide, kf, Sgt_spec), shading interp
    set(gca, 'Fontsize', 14, 'Xlim', [0 L], 'Ylim', [230 350])
    colormap(hot)
    xlabel('time (t)'), ylabel('Hz')
    title({'Piano Spectrogram', ['a = ', num2str(width(jj))]}, 'Fontsize',
30)

end
```

```
% Homework 2 - Gabor Transforms
% David Long
% 1.20.19

clear all; close all; clc

% Load signal

[yp, Fsp] = audioread('music2.wav');
Sp = yp.';
```

```
% Computational domain

L = length(Sp)/Fsp;
n = length(Sp);

t2 = linspace(0, L, n+1);
t = t2(1:n);

k = (2*pi/L)*[0:(n/2 - 1) (-n/2):-1];
ks = fftshift(k);
kf = ks/(2*pi);

b = L/3;
width = [128];

numSlices = 100;
tslide = linspace(0, L, numSlices);




% Signal function and transform

St = fft(Sp);

figure(1)
sgtitle('Signal And Frequencies', 'Fontsize', 30)

subplot(2, 1, 1)
plot(t, Sp, 'k')
set(gca, 'Fontsize', 14, 'Xlim', [0 L], 'Ylim', [-1 1])
xlabel('time (t)'), ylabel('S(t)')
title('Recorder Version', 'Fontsize', 20)

subplot(2, 1, 2)
plot(ks, abs(fftshift(St))/max(abs(St)), 'k')
set(gca, 'Fontsize', 14, 'Xlim', [-20000 20000], 'Ylim', [0 1])
xlabel('frequency (k)'), ylabel('FFT(S)')




% Dilation loop

for jj=1:length(width)

    Sgt_spec = [];

    % Translation in time
    for j=1:length(tslide)

        g = exp(-width(jj)*(t - tslide(j)).^2);
        Sg = g.*Sp;
```

```
        Sgt = fft(Sg);
        Sgt_spec = [Sgt_spec; abs(fftshift(Sgt))];

    end




    % Spectrogram of time-frequency signal

    figure(2)

    Sgt_spec = Sgt_spec.'/max(max(Sgt_spec));

    pcolor(tslide, kf, Sgt_spec), shading interp
    set(gca, 'Fontsize', 14, 'Xlim', [0 L], 'Ylim', [700 1100])
    colormap(hot)
    xlabel('time (t)'), ylabel('Hz')
    title({'Recorder Spectrogram', ['a = ', num2str(width(jj))]}, 'Fontsize',
30)

end




% Homework 2 - Gabor Transforms
% David Long
% 1.20.19

clear all; close all; clc

% Load signal

[yp, Fsp] = audioread('music1.wav');
Sp = yp.';




% Computational domain

L = length(Sp)/Fsp;
n = length(Sp);

t2 = linspace(0, L, n+1);
t = t2(1:n);

k = (2*pi/L)*[0:(n/2 - 1) (-n/2):-1];
ks = fftshift(k);
kf = ks/(2*pi);
```

```
b = L/3;
width = [128];

numSlices = 100;
tslide = linspace(0, L, numSlices);




% Step filter

klower = 1400;
kupper = 2200;
filter = zeros(1, length(Sp));

for j=1:length(filter)
    if ((ks(j) >= klower) && (ks(j) <= kupper))
        filter(j) = 1;
    end
end




Spf = ifft(ifftshift(filter.*fftshift(fft(Sp))));




% Dilation loop

for jj=1:length(width)

    Sgt_spec = [];

    % Translation in time
    for j=1:length(tslide)

        g = exp(-width(jj)*(t - tslide(j)).^2);
        Sg = g.*Spf;
        Sgt = fft(Sg);
        Sgt_spec = [Sgt_spec; abs(fftshift(Sgt))];

    end
```

```
    % Spectrogram of time-frequency signal

    figure(1)

    Sgt_spec = Sgt_spec.'/max(max(Sgt_spec));

    pcolor(tslide, ks, Sgt_spec), shading interp
    set(gca, 'Fontsize', 14, 'Xlim', [0 L], 'Ylim', [0 6000])
    colormap(hot)
    xlabel('time (t)'), ylabel('frequency (k)')
    title({'Filtered Piano Spectrogram', ['a = ', num2str(width(jj))]},
'Fontsize', 30)

end
```

C. References

[1] Kutz, J. N. February 26, 2010. *AMATH 582: Computational Methods for Data Analysis.* Department of Applied Mathematics, University of Washington, Seattle, WA 98195.

[2] Raissi, Rassol, December 2002. *The Theory Behind Mp3***.** Retrieved 23 January 2019, from https://www.mp3-tech.org/programmer/docs/mp3_theory.pdf

[3] *A Look at the Mathematical Origins of Western Musical Scales | The n-Category Café*. (2010). *Golem.ph.utexas.edu*. Retrieved 23 January 2019, from https://golem.ph.utexas.edu/category/2010/02/a_look_at_the_mathematical_ori.html

[4] *.:: TuxGuitar ::. Open Source Tablature Editor :: TuxGuitar*. (2019). *Tuxguitar.com.ar*. Retrieved 30 January 2019, from http://www.tuxguitar.com.ar/