Audio Signal Processing with Gabor Filtering, Wavelets, and Frequency Analysis

Joe Henthorn
University of Washington

January 2020

Abstract: The goal of this paper is to demonstrate how to filter a signal in time for better temporal resolution in frequency. This paper also explains how to visualize the Gabor filter method with a Gaussian and Mexican Hat filter, as well as spectrogram of the results.

1 Introduction

The following document is a guide for using the Gabor filtering method to clean a time signal. We use the Gaussian filter, the Mexican Hat filter, and the Shannon (step function filter) in the time domain (EQ:3) and (EQ:4). The accompanying MATLAB code is a guide for using time domain filters and the Fourier Transform to build spectrograms and to analyze audio signals in both time and frequency (EQ:1) and (EQ:2). In this example we use the built-in MATLAB audio "handle", and we also evaluate some audio recordings of a piano and a recorder. The MATLAB code also demonstrates how to make pretty plots and spectrograms of the time/frequency data.

2 Theoretical Background

The following formulas are the continuous Fourier Transform and the inverse Fourier Transform which are used to transform a time signal into its frequency elements.

$$Fourier \quad Transform$$
$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(x) \cdot e^{-2\pi i \omega} dx \tag{EQ:1}$$

For discrete values we use the discrete Fourier transform that use a sums of discrete points over a finite interval instead of an infinite integral. This is the equation we use for our numerical methods.

$$Discrete \quad Fourier \quad Transform$$
$$X_\omega = \sum_{n=1}^{N-1} x_n \cdot e^{-2\pi i \omega / N} d\omega \tag{EQ:2}$$

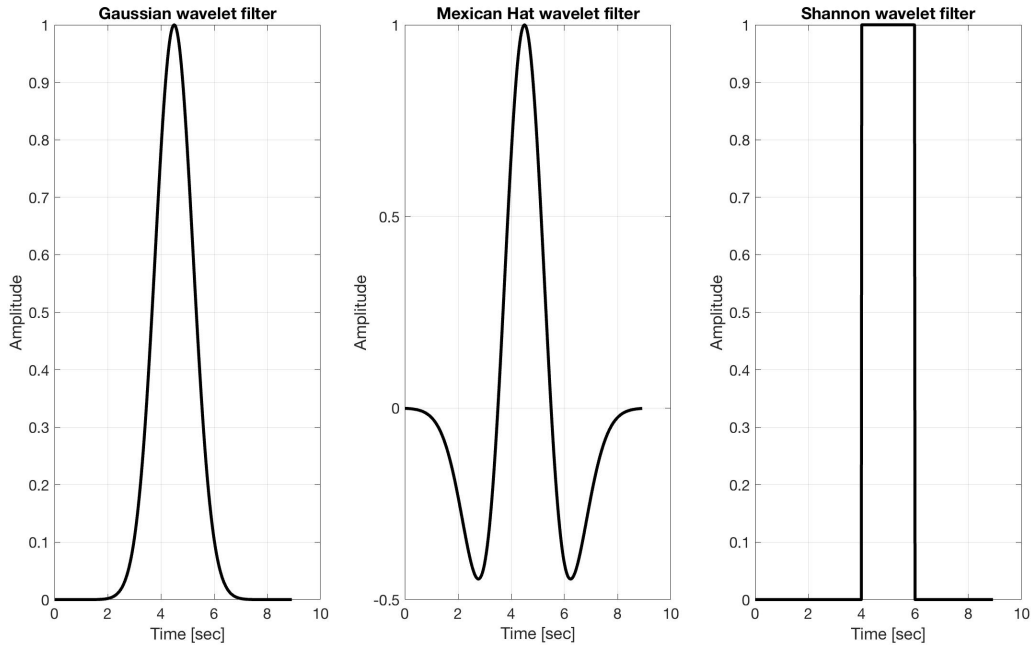
The Gaussian, Mexican hat, and Shannon wavelet filters (figure 1) are a mathematical tool that enables the preservation or increased resolution of temporal data in the frequency domain. The Gaussian filter is essentially a Gaussian curve (EQ:3) applied to a time domain signal. The Mexican hat wavelet filter (EQ:4) is a derivative of the Gaussain filter and has better time localization properties.

$$Gaussian \quad Filter \quad Function$$
$$\mathcal{G}(t) = e^{-\tau (t - t_0)^2} \tag{EQ:3}$$

$$Mexican \quad Hat \quad Filter \quad Function$$
$$\psi(t) = (1 - (t - t_0)^2 \cdot e^{-\tau(t-t_0)^2} \tag{EQ:4}$$

3 Methods & Algorithm Implementation

For this example, we use hard coded the guassian and mexican hat filters in matlab. We also used the built in MATLAB function unitstep. Further, we used the matlab command **fftn()** and **ifftn()** to transform the audio data with the Fourier transform (EQ:1) & (EQ:2). In this particular example, we visualize the filtering process in time and frequency. First the time domain data is loaded and its time length is determined by multiplying the sample rate by the number of samples. A frequency vector is created for our dependant variable axis in the frequency domain, and it is scaling by the factor π/L, where L is the number of data points. The data vector is then multiplied by our filter and then the signal data is transformed with the **fftn()** function (EQ:2). This is done over a for loop for each sample of data. Inside our for loop, we collect every Fourier transform of our data into a column vector that will build our spectrogram. Lastly, we use the **pcolor()** command to plot our spectrogram. We also use **colormap(hot)** because we like it hot and spicy. Additionally, the value we pick for τ determines the width of our filters. Lower values of τ produce wider filter windows, while higher values produce a narrow filter window. We also try oversampling with small translations of our filters across our audio signal, and we try under-sampling with large translations of our filters across our audio signal and we visualize the results.
Another particularly useful method for this exercise is to down sample our large piano and recorder audio files we achieve this with the **rat()** function in MATLAB (See code).

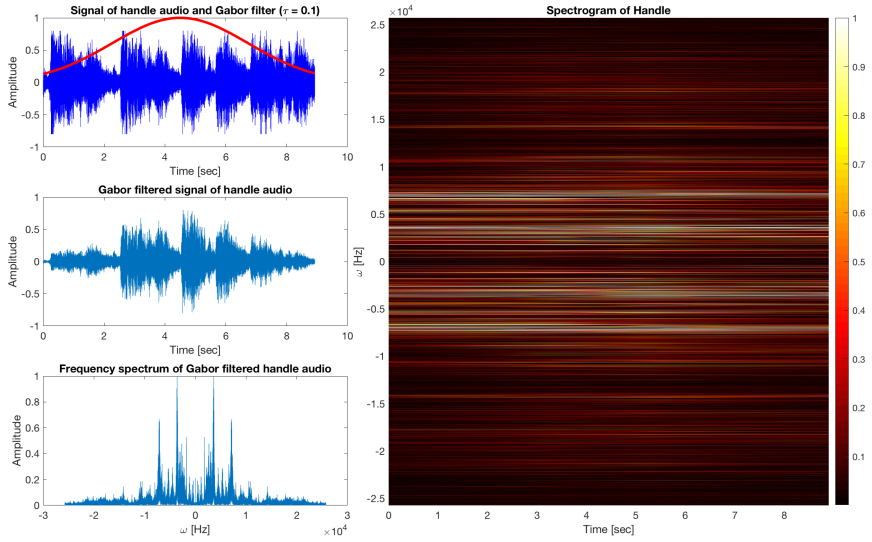


.

Figure 1: Plot of Gaussian function, Mexican hat function, and the Shannon step function filters. All filters centered in the middle the handle audio signal.

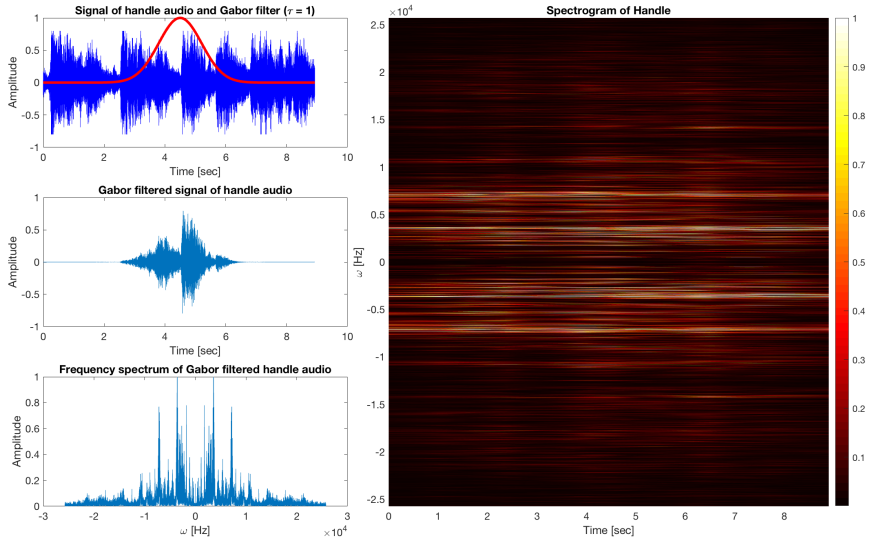
4 Computational Results

In this example we compare large and small values for τ(filter width). We compare the Guassian vs Mexican Hat filter, and we also compare undersampling and oversampling.



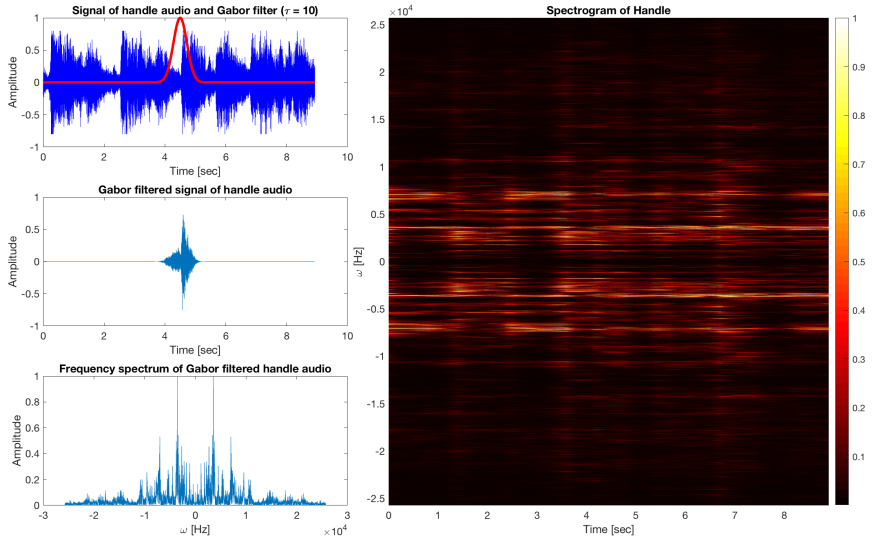
.

Figure 2: Top plot shows audio signal in time and the Gaussian filter with a tau = 0.1. Middle plot shows Gabor filtered audio signal in time. Bottom graph shows.



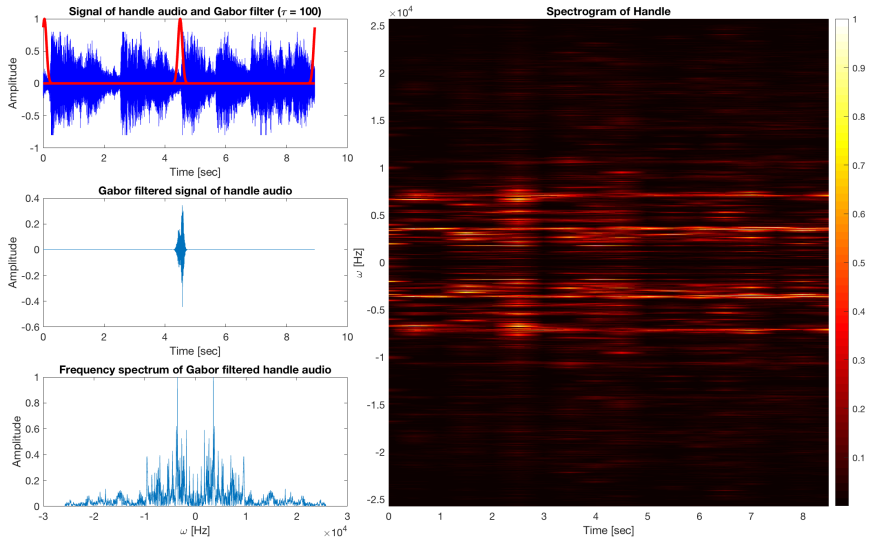
.

Figure 3: Top plot shows audio signal in time and the Gaussian filter with a tau = 1. Time step = 0.8925. Middle plot shows Gabor filtered audio signal in time. Bottom graph shows filtered frequency at t = 4.5 sec.



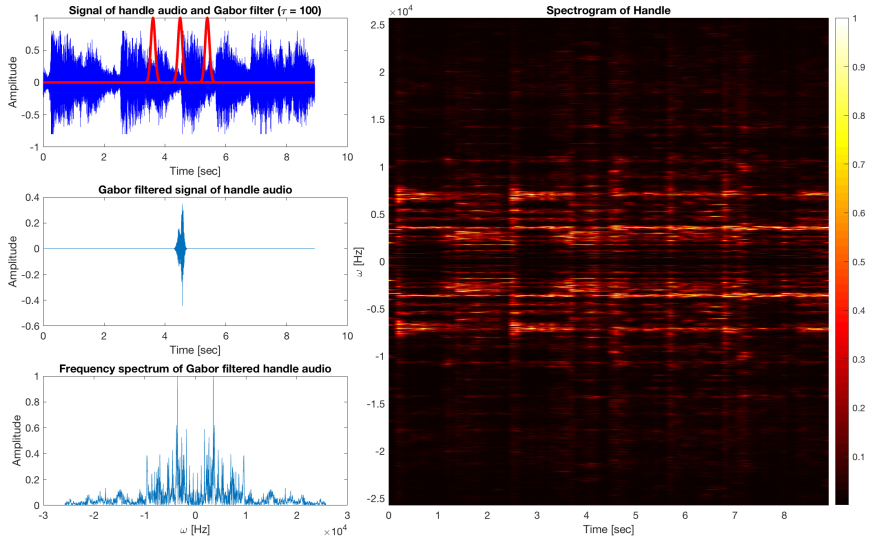
.

Figure 4: Top plot shows audio signal in time and the Gaussian filter with a tau = 100. Time step = 0.8925. Middle plot shows Gabor filtered audio signal in time. Bottom graph shows filtered frequency at t = 4.5 sec.



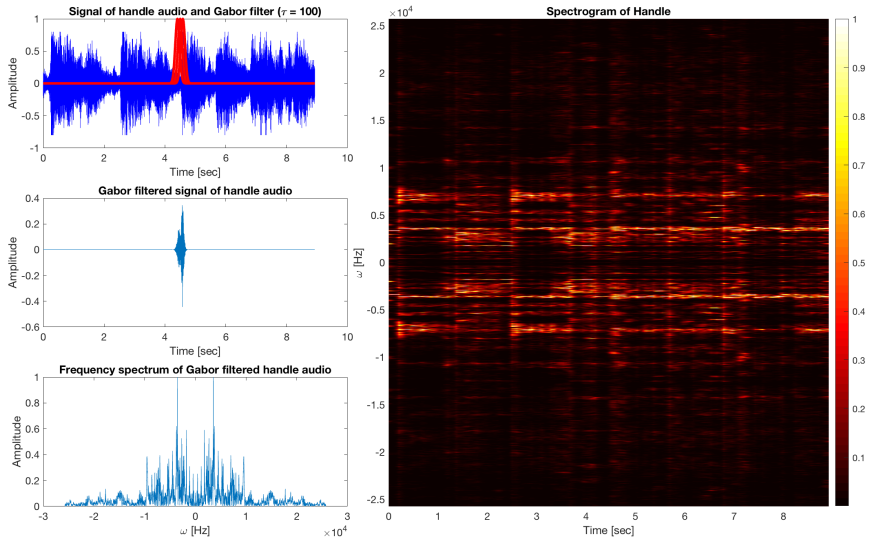
.

Figure 5: Top plot shows audio signal in time under sampling with the Gaussian filter with a tau = 100. Three translations are depicted. Time step = 4.4625 sec shift. Middle plot shows Gabor filtered audio signal in time. Bottom graph shows filtered frequency at t = 4.5 sec.



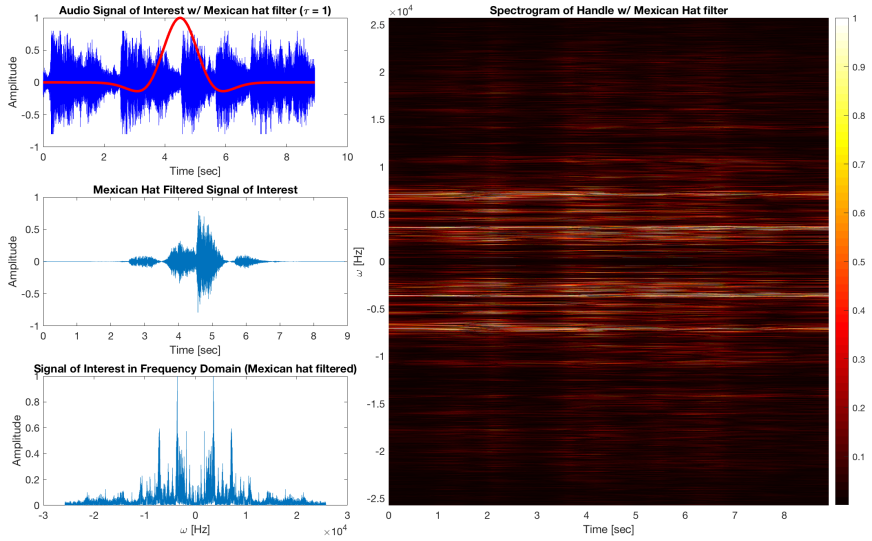
.

Figure 6: Top plot shows audio signal in time medium sampling with the Gaussian filter with a tau = 100. Three translations are depicted. Time step = 0.8925 sec shift. Middle plot shows Gabor filtered audio signal in time. Bottom graph shows filtered frequency at t = 4.5 sec.



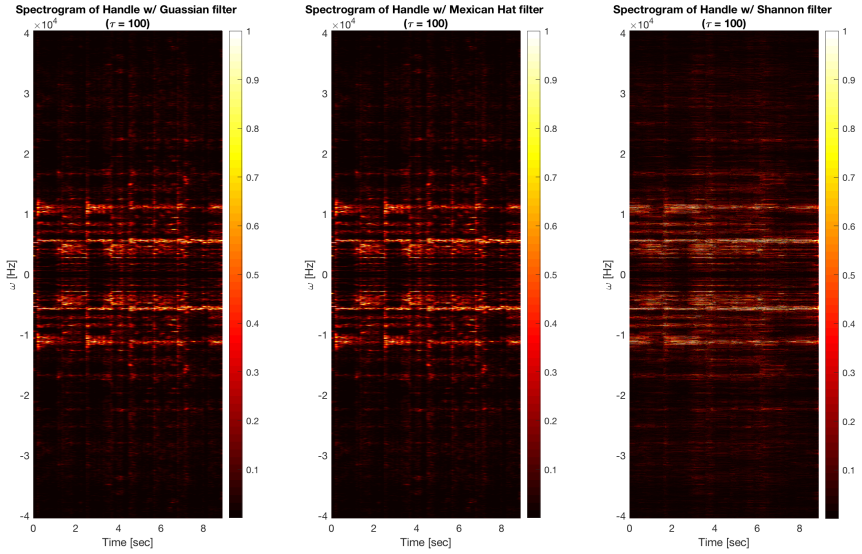
.

Figure 7: Top plot shows audio signal in time over-sampling with the Gaussian filter with a tau = 100. Three translations are depicted. Time step = 0.08925 sec shift. Middle plot shows Gabor filtered audio signal in time. Bottom graph shows filtered frequency at t = 4.5 sec.



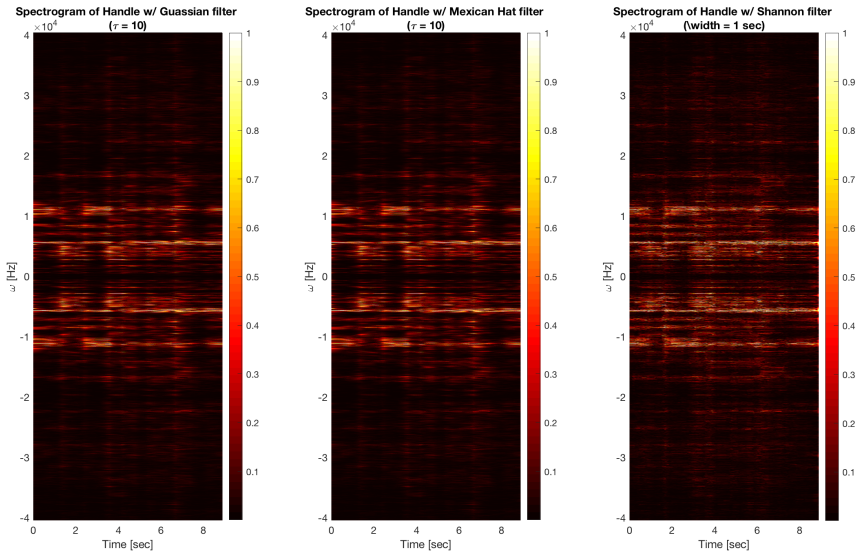
.

Figure 8: Top plot shows audio signal in time medium sampling with the Mexican Hat filter window with a tau = 100. Three translations are depicted. Time step = 0.8925 sec shift. Middle plot shows Gabor filtered audio signal in time. Bottom graph shows filtered frequency at t = 4.5 sec.



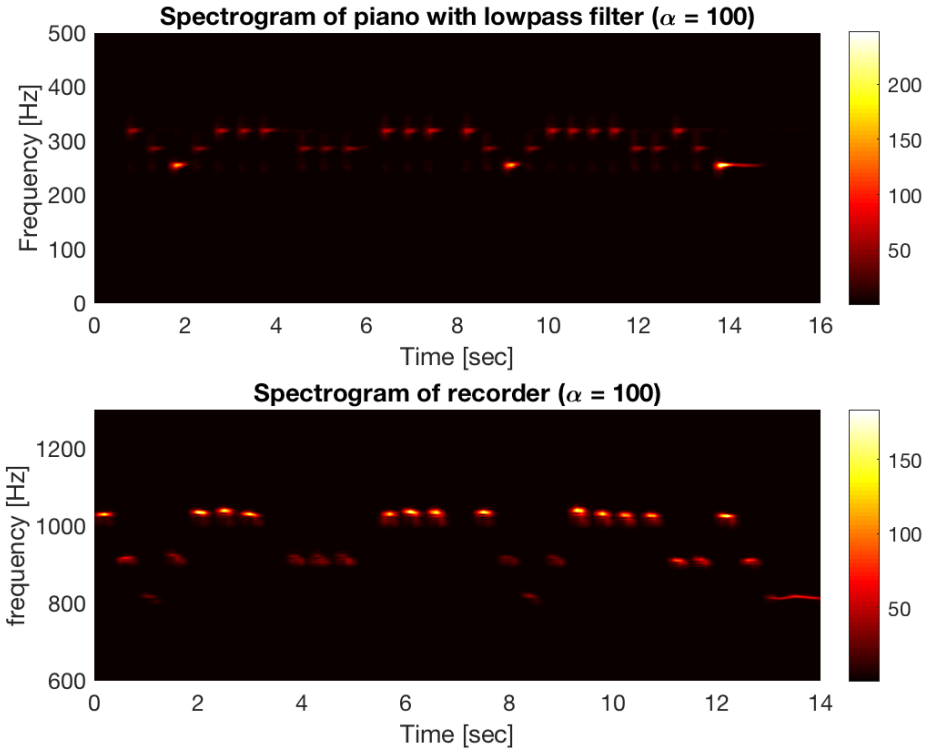
.

Figure 9: Plot of undersampled spectrograms (4.0925 sec shift window) for Gaussain, Mexican Hat, and Shannon wavelength window. Shannon wavelet demonstrates the best frequency resolution. α= 100



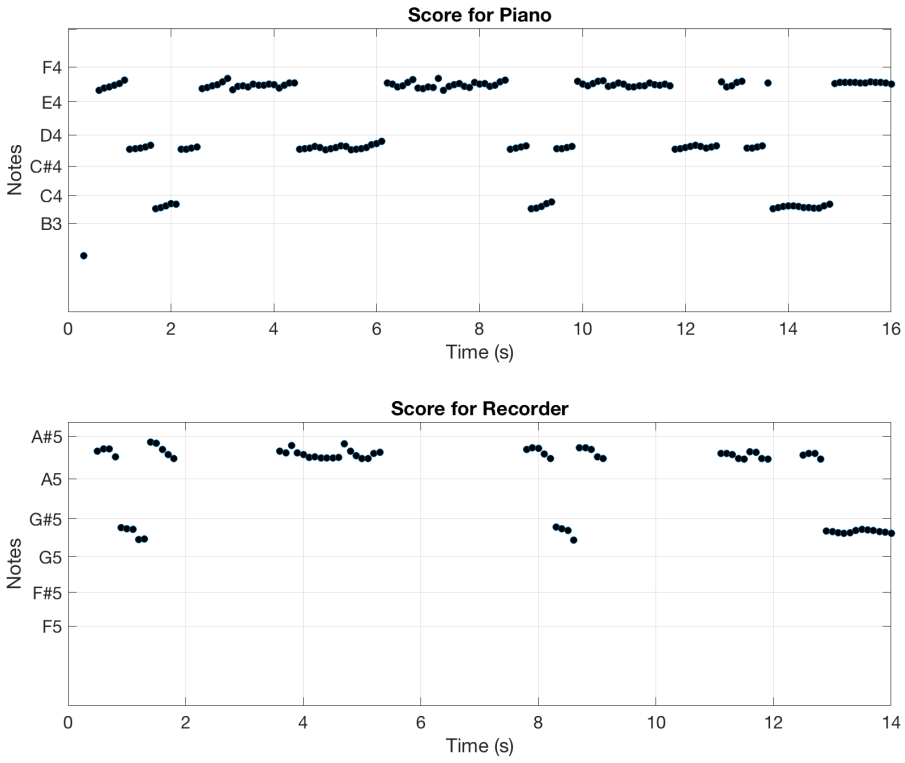
.

Figure 10: Plot of oversampled (0.08925 sec shift window) spectrograms for Gaussain, Mexican Hat, and Shannon wavelength window. Shannon wavelet demonstrates the best frequency resolution. α= 100



.

Figure 11: Plot of piano and recorder spectra with colormap hot spectrograms. Piano overtones were filtered with a low-pass filter.



.

Figure 12: Plot of the recreated score from the piano and recorder audio files.

5 Conclusion

In this example we used audio signal data and cleaned it up in the time domain. We used Gabor filtering with a Gaussian filter, Mexican Hat filter, and a Shannon filter to clean the time domain signal. We used the Fourier transform to get the frequency representation of our data. We also built up a spectogram with this data to visualize the time and frequency content at the on the same plot (figure 2-10).
Here we found that the best Gabor filter was the mexican hat wavelet which out performed than the Gaussian and Shannon filter. This is shown in figures 9 10. representations. We also found that higher values of τwe picked for our filters improved our temporal resolution (figure 2-10). Furthermore, found that oversampling improved our frequency resolution (figure 7 10).
Lastly, we used the Gabor Gaussian filter to clean up two recording of "Mary Had a Little Lamb." We also used a low-pass filter in frequency space to clean up the signal and relate it to recreated score (figures 11 12).