AMATH 582 Homework 2: Gábor Transforms and Music

Eric A. Silk

***Abstract*—This exercise served as an opportunity to explore the implementation of a Spectrogram, along with the various considerations that go into window selections and their respective parameters. In addition, this was then applied to music to reproduce a musical score of two pieces of music provided by Dr. Kutz.**

***Index Terms*—Filter, Fourier, Gabor, STFT, Spectrogram, Window, Audio**

I. Introduction

GÁbor transforms, or ”short time Fourier transforms” (STFT’s) are an incredibly common technique used in signal processing and analysis. Their primary advantage is that they trade some resolution within the frequency domain for improved resolution in the time domain, or ”time localization.” This is typically acceptable, allowing end-users to understand and visualize signals with frequency components that may start, stop, or modulate throughout the duration of the data.

II. Theoretical Background

A. Fourier Transforms

The Fourier Transform has already been covered in greater depth in the first paper, but is reproduced here for reference (Equation 1). In short, it converts arbitrary functions into a series of sines and cosines, which can be used to extract information about periodic behaviors in the input function.

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(x) e^{-2\pi j x \omega} dx \qquad (1)$$

B. Gábor Transforms

The Gábor transform is defined in Dr. Kutz’s book and is reproduced in (2). The addition of the windowing function $g(\tau)$ is what provides for the time localization. By sliding this window across the data, sections of the signal are isolated, providing temporal localization.

$$G[f](t,\omega) = \hat{f}_g(t,\omega) = \int_{-\infty}^{\infty} f(\tau) g(\tau - t) e^{-j\omega\tau} d\tau \qquad (2)$$

Eric Silk is a Masters Student in Applied Mathematics at the University of Washington, and a Research Engineer for Schweitzer Engineering Laboratories, Pullman, WA 99163 (email: esilk16@uw.edu, eric.silk@ericsilk.com)

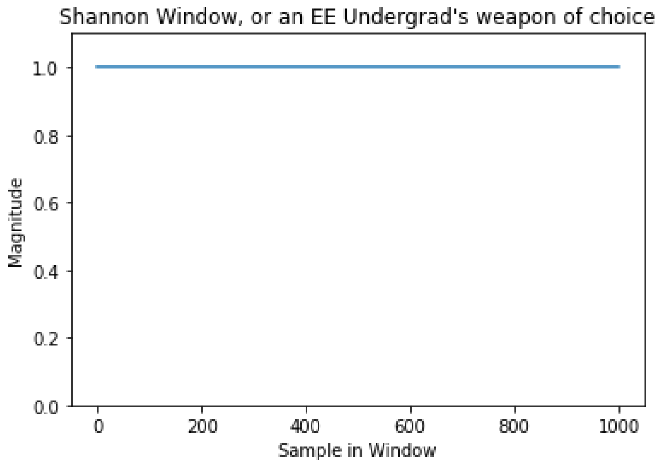


Fig. 1. The Shannon, Boxcar, or Rectangular Window

C. Windowing Function

The selection of the windowing function provides several advantages, with several considerations. Firstly, it is what provides the temporal localization (as noted above). Secondly, if selected such that it goes towards zero at its edges, it can provide a reduction in spectral leakage and artifacts. These normally occur due to the assumption that the function is periodically repeated, which may result in sharp transitions. Sharp transitions in a signal correspond to higher frequency components when transformed to the Fourier domain.

The considerations for window selection are beyond the scope of this paper, but can be surmised in three parameters:

- Primary lobe width, or the selectivity of the transform
- Initial side-lobe height
- Side-lobe attenuation rate, or how quickly the leakage decays

More information can be found on the relevant Wikipedia Page.

Several choices include the Shannon (1), Gaussian (2), Triangular (3), and ”Mexican Hat” (4) window.

III. Algorithm Implementation and Development

A. Gábor Transform/Spectrogram

In the discrete space, the Gábor transform involves three steps:

1) Signal segmentation
2) Application of the windowing function
3) The discrete Fourier transform

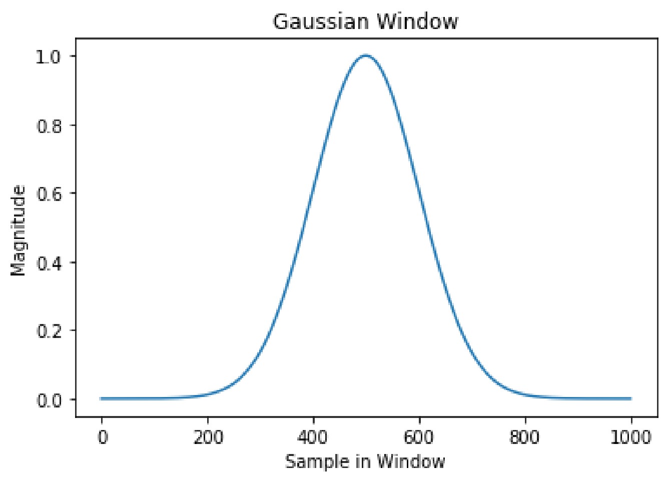


Fig. 2. The Gaussian Window

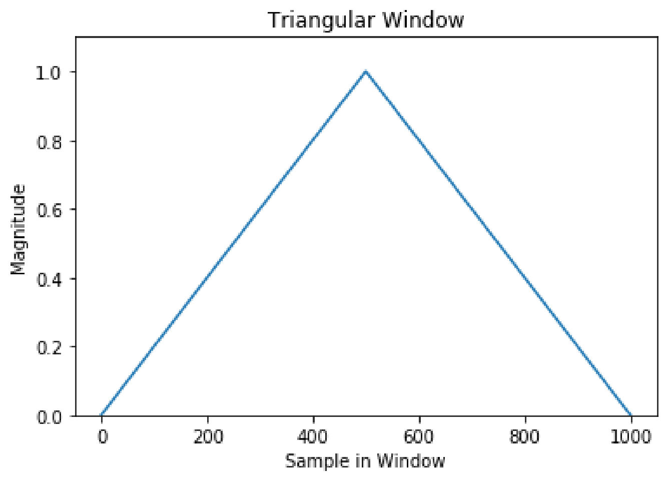


Fig. 3. The Triangular Window

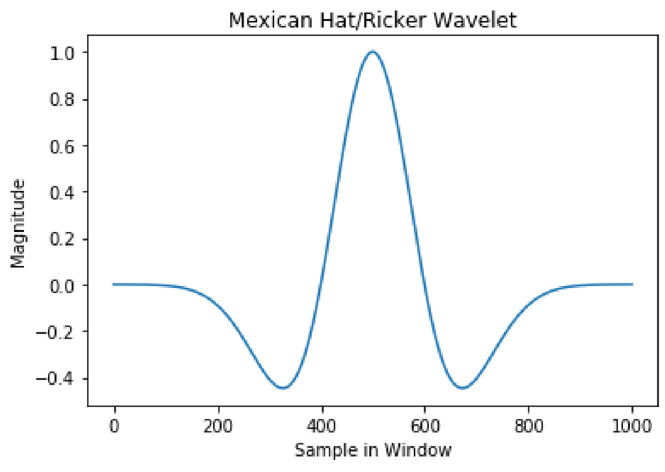


Fig. 4. The Ricker, or Mexican Hat Wavelet

The first of these is achieved through stride manipulation. "Stride" refers to the number of addresses to shift to increment by one element in an array. For instance, if the smallest representable type is 8 bits, and the data is stored as a 64 bit type, the stride of the array is $64/8 = 8$, or 8 bytes per stride. However, this can be intentionally set to a longer multiple of the native stride to step multiple elements at a time. NumPy provides a method for this, given a stride length and width, to return a "view" into an array.

Once a windowed view is constructed, each window can be iterated over (in a naive implementation) and multiplied element-wise with a windowing function of an equivalent size/shape. That is, if a window is N samples long, the windowing function must also be N samples long. If no windowing function is applied, this is equivalent to the application of a Shannon window.

Windowing functions must always be bounded for this reason. Continuous functions are truncated; for instance, the Gaussian Window is defined as:

$$w[n] = exp(-\frac{1}{2}(\frac{n - N/2}{\sigma N/2})^2), 0 \leq n \leq N \qquad (3)$$

Each resulting windowed function can then have the DFT taken via the Fast Fourier Transform (FFT), and the results stored. In this case, the data was purely real, and thus the "real" FFT (or RFFT) was used, automatically discarding the negative frequency components.

In order to plot the resulting spectrogram, two other pieces of information are required: the time of each window, and the frequencies contained therein. See the `spectrogram()` method for details in `fft_funcs.py` for details.

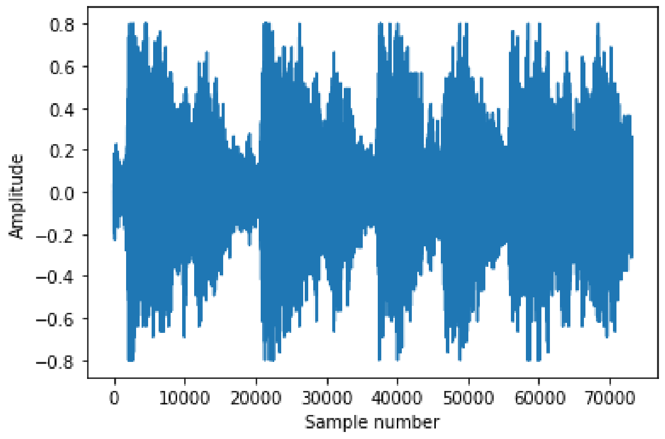
B. Filtering and Score Reproduction

Given a single sound source, such as one voice or one instrument, a spectrogram can be used as a method to produce a musical score of sorts, analogous to the way MIDI encodes musical information. In order to clean the data up, filtering must be used. Rather than rely on convolution in the Fourier domain (or a piecewise multiplication of the transformed signal and the transformed filter), a forward-backwards filter was used. These are typically computationally more robust, and the use of forward-backwards ensures linear phase (beneficial when dealing with audio signals). Knowing that a note is typically a fundamental combined with overtones, a bandpass filter was used to remove sub- and super-fundamental frequencies. The cutoff frequencies were determined via visual inspection and manually tuned.

When the filtered signal is then converted to a spectrogram, the largest magnitude values in the spectrum can be taken as the pitch of the note during a given window.

IV. Computational Results

Firstly, we can see the plot of the `handel.mat` file in IV. Note that it appears highly dense and varied across the entirety of the time series. Interestingly, with a priori knowledge of the music, one can see how the envelope of the signal corresponds to the overall loudness and sustain of the singers.



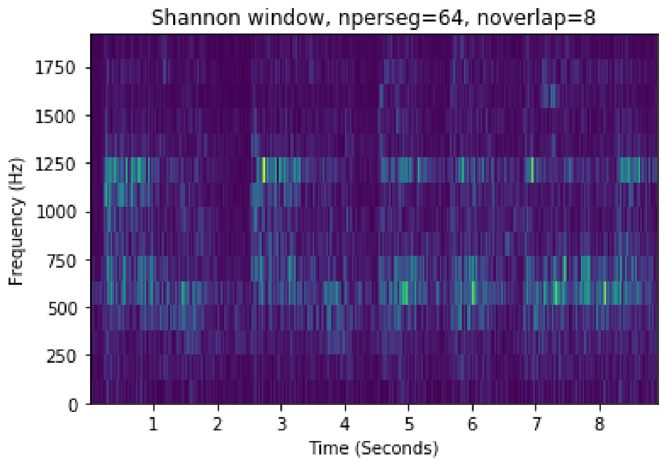


Fig. 5.

A. Varying Window Dimensions and Overlap

The Shannon window was selected for varying the window size and overlap as it is a “trivial” window. As expected, small windows produce data that is poorly localized along the frequency axis, but highly localized along the time axis. Longer windows reverse this, producing highly localized data along the frequency axis and poorly localized temporal data. (5)(6)(7)

By varying the amount of overlap while holding the window the same, we can observe its effects. It is difficult to tell visually, but longer overlaps trade greater computational time for an increased ”averaging” effect between windows, decreasing the effects of the windowing.(8)(9)(10)

B. Different Windowing Functions

Differing windowing functions given different behavior within the frequency domain as well. Certain functions are more selective in frequency at the expense of reduced SNR (signal-to-noise-ratio), others are the opposite. Examples explored include the triangular window (11), Gaussian window (12), and Ricker or “Mexican Hat” window (13).

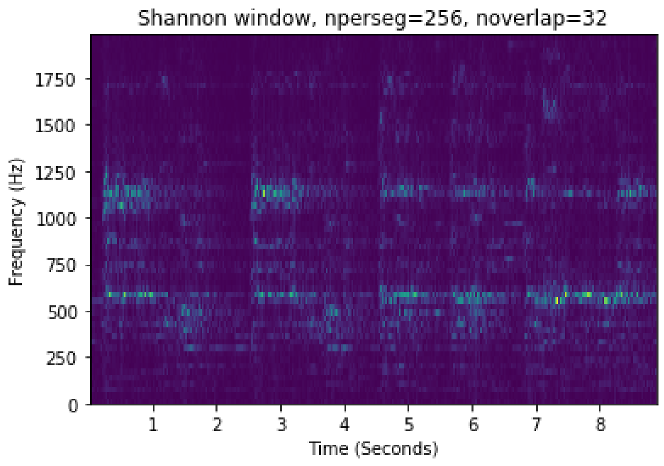


Fig. 6.

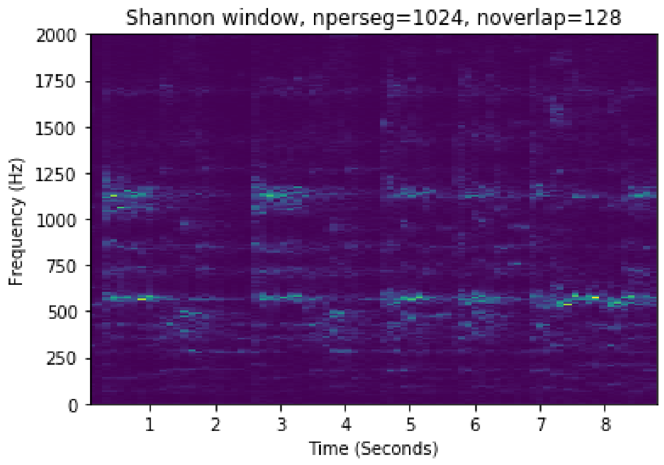


Fig. 7.

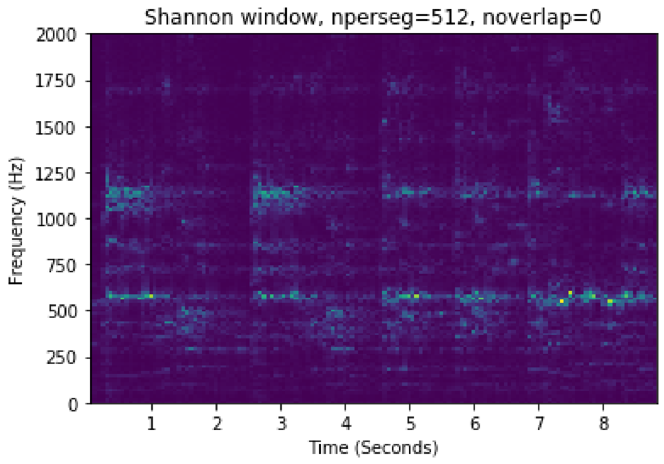


Fig. 8.

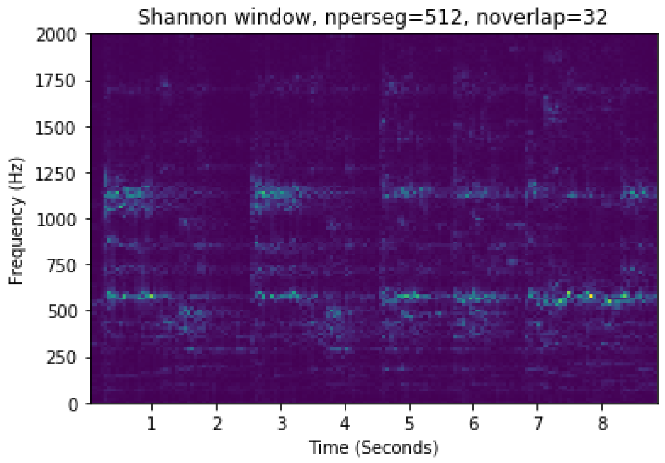


Fig. 9.

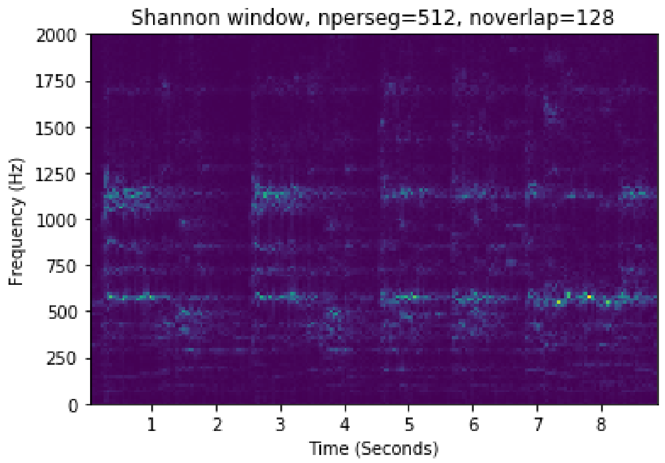


Fig. 10.

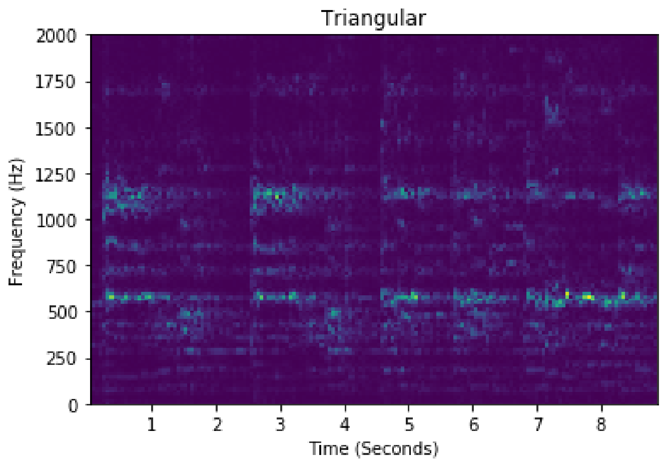


Fig. 11.

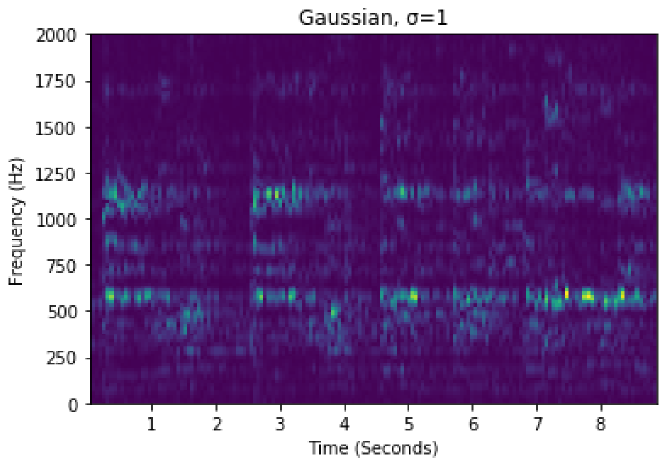


Fig. 12.

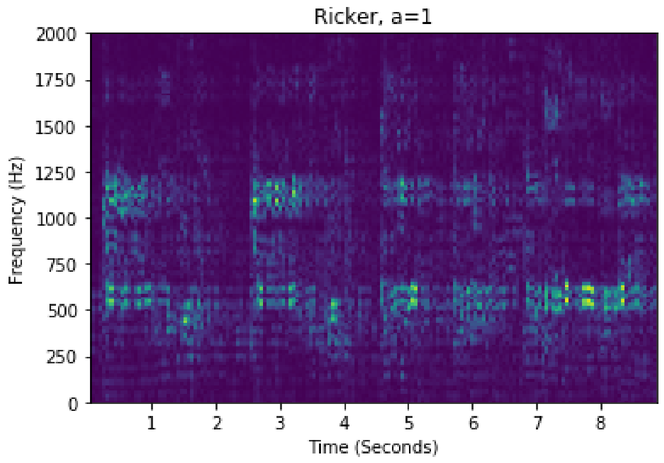


Fig. 13.

C. *Score Reproduction*

The time series of the piano(14) and recorder(15) recordings are reproduced for visual inspection. Notably, the recorder is clearly louder (likely due to proximity to the microphone), and the envelope of each note indicates more sustain – common for instruments with continuous excitation of the vibrating member of the instrument. This is in contrast to instruments where excitation may only occur once, such as the piano or a guitar, where the vibrating member is struck once and initial transient decays quickly.

Taking the spectrogram, and normalizing the results to a log scale to emphasize the harmonic content, the spectral energy of the sound can be observed.

Of particular note is the relatively strong harmonics of the recorder when compared to the piano. High levels of harmonics are typically associated with strong non-linearities in a system. The recorder also seems to exhibit predominantly odd harmonics, typical of a symmetric distortion. Both of these qualities are often described qualitatively as ”harsh” or ”brash”. For more examples, listen to any Metal record – the distinctive timbre of the guitars is achieved through

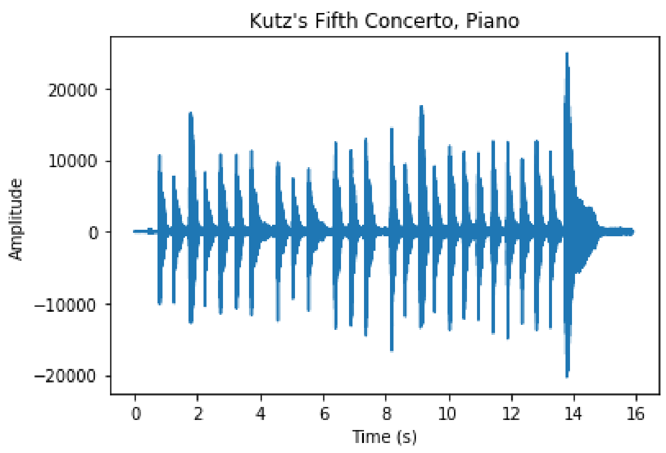


Fig. 14.

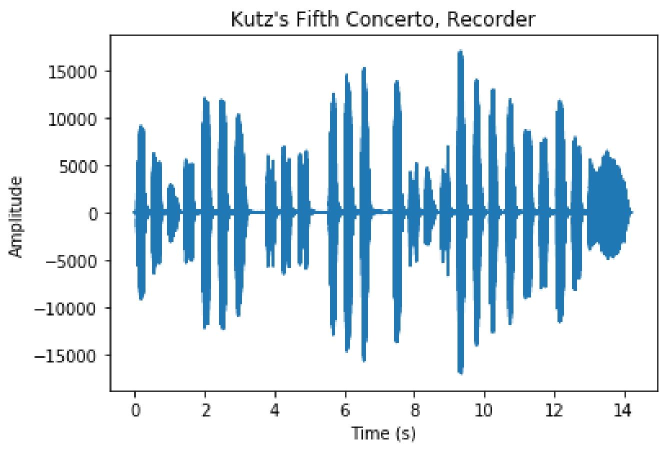


Fig. 15.

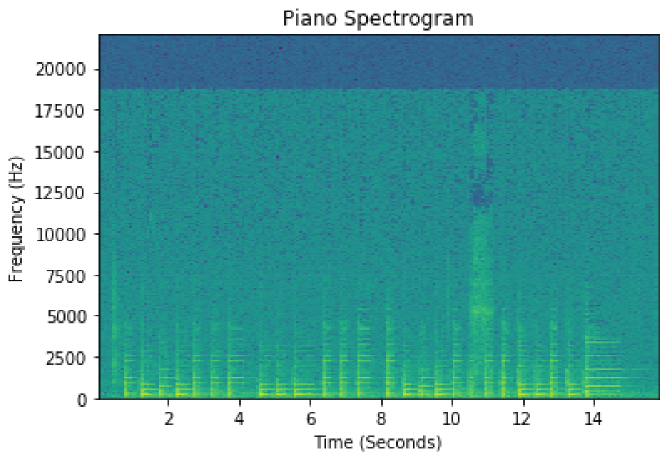


Fig. 16.

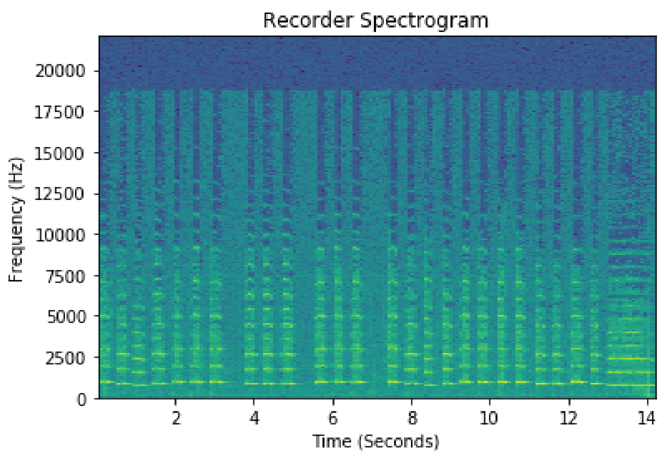


Fig. 17.

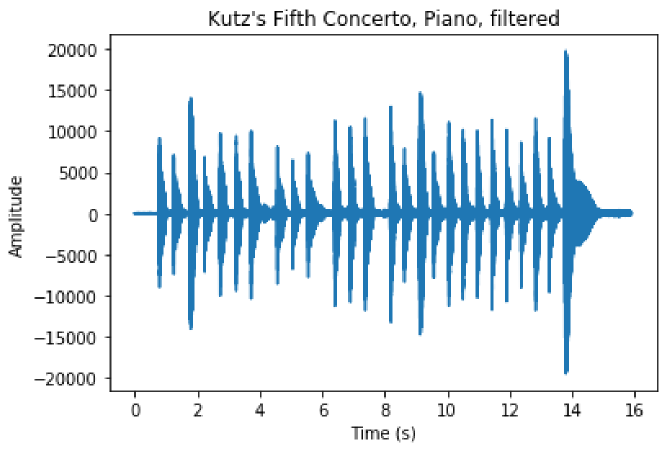


Fig. 18.

the intentional introduction of significant non-linearity in the signal path.

Conversely, the piano appears to have a much more complex harmonic spectrum, owing likely to the presence of even harmonics.

The sudden dropoff in spectral content above 18kHz is likely due to an anti-aliasing filter on the recording device.

Both signals were bandpass filtered with cutoffs appropriate to their respective pitches, and the resulting time series can be seen in 18 and 19.

The resulting spectrograms (and an approximate musical score) are shown in 20 and 21.

V. Summary and Conclusion

Gábor filtering proves to be a highly useful analytic technique when assessing and processing time series data. By allowing a trade-off between time and frequency localization, a compromise can be made to suit the data. In addition, with the addition of filtering, a rough pitch detection and score reproduction method can be implemented.

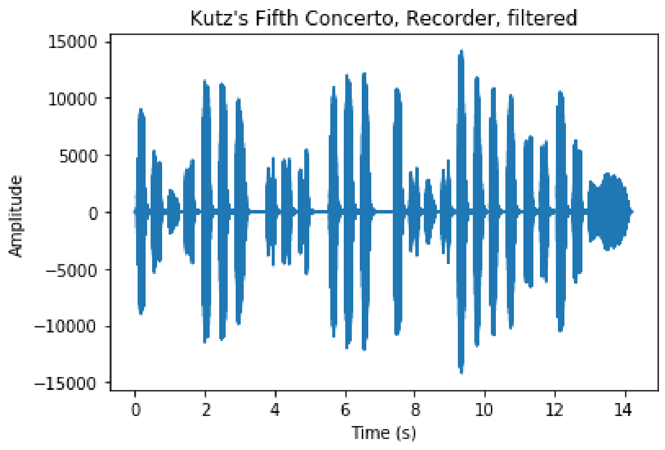


Fig. 19.

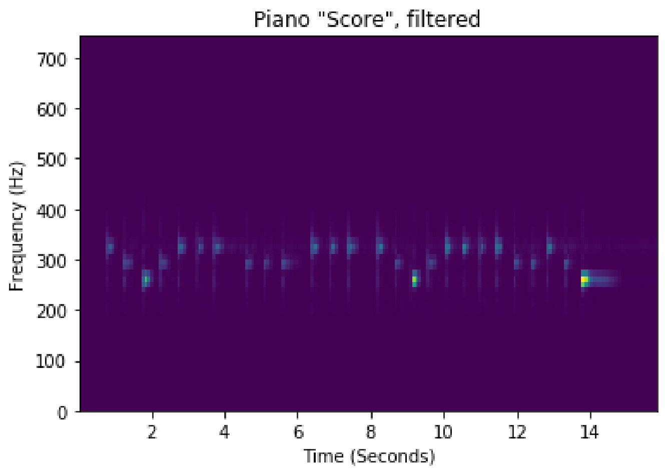


Fig. 20.

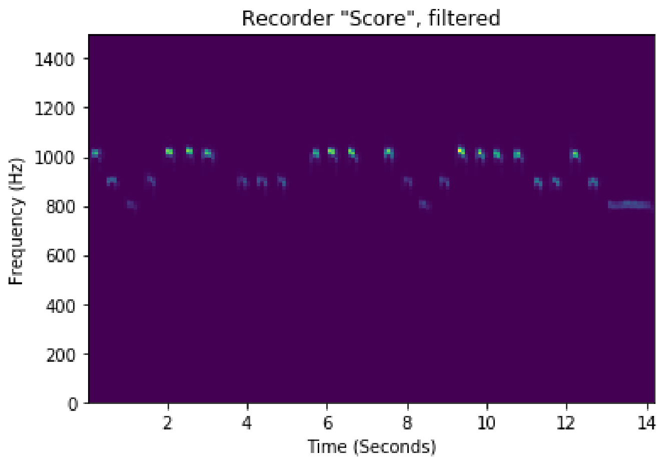


Fig. 21.

VI. Appendix A: Functions Used

A. `numpy.lib.stride_tricks.as_strided()`

Given a dataset (1D, in this case), a stride length, and a window width, return a "view" into the data. See here.

B. `numpy.arange()`

Generates a linearly spaced, half-open set of values in a vector. Allows specification of the start, stop, step size, and data type. See here.

C. `numpy.unique()`

Returns the unique elements of an array, sorted. Used to remove the duplicate values produced when calculating the frequencies in `spectrogram`. See here.

D. `numpy.apply_along_axis()`

Applies a function (provided as a function object) to an `ndarray` along a given axis. Used to calculate the `RFFT` of each window of the data. See here.

E. `numpy.fft.rfft()`

Computes the 1D DFT on real input, returning only the positive frequencies. Conveniently handles the discarding of negative frequencies and removes the need for a post-FFT shifting function. See here.

F. `numpy.where()`

Returns the indices of values matching the logical condition passed as an argument. See here.

G. `matplotlib.pyplot.pcolormesh`

Given a pair of axes and a "mesh" of values for those two axes, produces a color coded plot indicating their magnitudes. See here.

H. `matplotlib.colors.Normalize()`, `matplotlib.colors.LogNorm()`

For use in conjunction with a color plot, normalizes the colors to extend from the desired minimum and maximum values (typically the minimum and maximum of the data). LogNorm does the same, but on a logarithmic basis, useful for showing small differences. See here.

I. *Windowing Functions*

SciPy provides easy methods for generating windowing functions, including Gaussian, Ricker, and Triangular.

VII. Appendix B: Python Code

See my Github for the full repository, including this source code for this IEEE template LaTeXdocument. Code is also attached at the end of this report.

VIII. Appendix C: Other Things

As a fun aside, the composer for 2016 reboot of "Doom", Mick Gordon, gave an excellent talk on the techniques and values drawn upon during his writing and recording process. It can be viewed here (approx. 1hr). In it he discusses an Easter Egg he embedded within the music – in particular, a manipulation of the spectrogram, producing the spectrogram in 22 (demonstrated around the 39 minute mark). Neat!

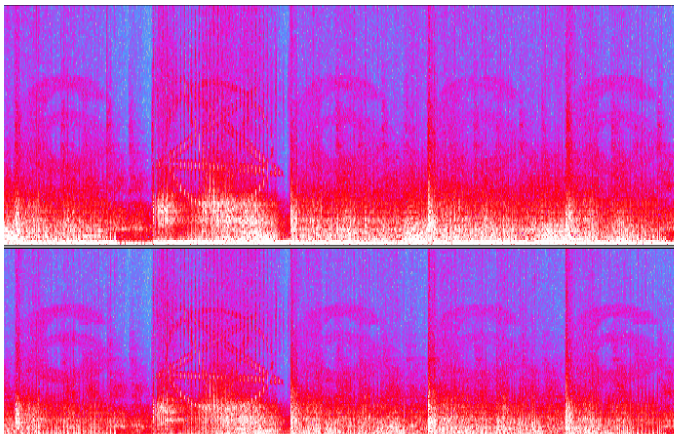


Fig. 22. Satanic Spectrograms in Doom