Analysis of Music Signals with Gabor Transform

Arvindh Sharma [*]

February 15, 2020

Abstract

Gabor transform was used to study the evolution of frequency composition with time in music clips. Analysis of a recording from Handel's "Messiah" showed the frequencies pertaining to the various people and instruments generating the music, and the instances at which the sounds were generated. Filter construction and its the translation of Gabor window were varied to explore the effects on the resulting spectrograms. In another application, records of the rhyme *Mary had a little lamb*, played with a piano and a recorder, were analyzed to study the difference between the instruments. The fundamental harmonics were identified and the overtones were filtered out to reproduce the music score.

1 Introduction and Overview

Fourier transformation is a useful tool to disassemble a signal into its constituent frequencies. However, the transformation analyzes the signal as whole – meaning that the frequency domain information pertains to the entire signal and it is not localized to a particular segment of the signal. Therefore, there is no way to pinpoint the temporal location (or spatial location in spatial data) at which a particular frequency is detected. This is especially problematic in analyzing non-periodic signals that do not repeat, such as music signals. Short Time Fourier Transform (STFT), or *Gabor transform*, helps overcome this shortcoming and allows us to determine how the frequency content evolves in a signal. This work illustrates the power of Gabor transforms by applying the technique to music signals, a clip from Handel's "Messiah", and recordings of the nursery rhyme *Mary had a little lamb* from a piano and a recorder, and analyzing the resulting spectrograms.

2 Theoretical Background

The Gabor transform is a modification of the Fourier transform, and it enables us to get the frequency content of a signal at different points along the signal's evolution. This is achieved with the help of Gabor windows, which are modifications to the Fourier transform kernel, which allows us to deduce the frequency content in a small window of the signal. By translating the window across different point in the signal, the frequency domain information in different segments of the signal can be evaluated [1]. The Gabor kernel is given by Eq. (1), where the parameter t determines the center of the kernel as it translates across the signal.

$$g_{t,\omega}(\tau) = e^{i\omega t} g(\tau - t) \quad . \tag{1}$$

The Gabor transform is then given by Eq. (2), where $\overline{g}$ is the complex conjugate of g, and its inverse is given by Eq. (3).

$$G(f)(t,\omega) = \hat{f}_g(t,\omega) = \int_{-\infty}^{\infty} f(\tau) e^{-i\omega t} \overline{g}(\tau - t) d\tau = \langle f, g_{t,\omega} \rangle \quad . \tag{2}$$

$$f(t) = G^{-1}\big(\hat{f}_g(t,\omega)\big) = \frac{1}{2\pi \|g\|^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \hat{f}_g(\tau,\omega) g(t-\tau) e^{i\omega t} d\omega dt \quad . \tag{3}$$

The width of the kernel $g(\tau - t)$ limits the range of frequencies that can be detected by the transform. This is so, since a shorter window would be unable to pick up longer wavelength signals since their periods would be bigger than the window width. On the other hand, such a window would provide better localization in terms of the time

[*]GitHub : `https://github.com/arvindhsharma`

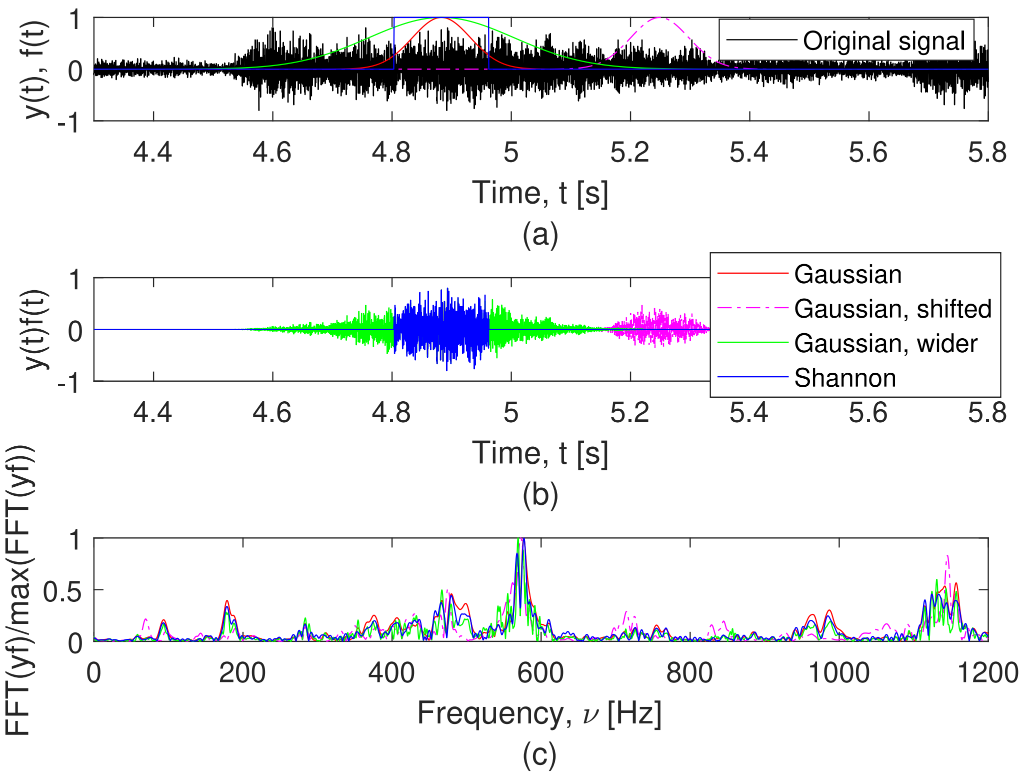

Figure 1: (a) A sample signal with different Gabor windows imposed on it; (b) Original signal modified by the Gabor kernels; (c) FFTs of the windowed signal.

at which the frequencies in the window are detected. This feeds into the ”uncertainty principle” associated with the transformation.

There could be many choices for the Gabor kernel. For instance, Eq. (4a) and Eq. (4b) are Gaussian and Ricker (Mexican Hat) kernels, respectively, with a width determined by the parameter σ. Eq. (4c) is a step-function (Shannon) filter of width $2a$, while Eq. (4d) is a custom triangular filter of width $2a$ as well. All of the filters in Eq. (4) are centered at τ, and by changing its location, the window of the STFT can be moved across the signal. Similarly, the widths of the kernels can be adjusted to get the frequency range of interest, knowing that there will be a concurrent trade-off in temporal resolution. Fig. 1 illustrates how the filter construction affects what frequencies are detected in the window, while also demonstrating how the translation of the window helps look at the frequency content at different points along the signal.

$$g(t) = e^{-(t-\tau)^2/(2\sigma^2)} \quad , \tag{4a}$$

$$g(t) = \frac{2}{\sqrt{3\sigma}\pi^{1/4}}\left(1-\left(\frac{t-\tau}{\sigma}\right)^2\right)e^{-(t-\tau)^2/(2\sigma^2)} \quad , \tag{4b}$$

$$g(t) = \begin{cases} 1, & \text{if } t \in [\tau-a, \tau+a] \\ 0, & \text{otherwise} \quad , \end{cases} \tag{4c}$$

$$g(t) = \begin{cases} \frac{t-(\tau-a)}{a}, & \text{if } t \in [\tau-a, \tau] \\ -\frac{t-\tau}{a}, & \text{if } t \in [\tau, \tau+a] \\ 0, & \text{otherwise} \quad . \end{cases} \tag{4d}$$

Once the frequency domain information is available at different instances along the signal of interest, a spectrogram can be constructed, which is a way to display the frequency content as it evolves along the length of the signal. Since we implement Gabor transform numerically in computers, the frequency information will be sampled at discrete intervals along the signal. Therefore, the amount of translation between successive windows plays a role in how much data is contained in the spectrogram. For short window traversals, the data density will be high and the processing time will be high, while it data density and processing times will be low for larger traversals.

3 Algorithm Implementation and Development

3.1 Handel's "Messiah"

Algorithm 1 has been developed to analyze a piece of music from Handel's "Messiah." Here is a brief overview of the process.

Algorithm 1 Algorithm to analyze the spectral signatures in Handel's "Messiah."

```
1: Import data for Handel's music piece from Matlab
2: $y \leftarrow$ Time-series data
3: Define time and Fourier domains, and visualize in both domains
4: procedure SPECTROGRAM($\Delta t, \sigma(optional)$)        ▷ To create a spectrogram with given input parameters
5:     for $ii = 1 : 1 :$ length ($\Delta t$) do
6:         $g \leftarrow$ Filter kernel
7:         $yg \leftarrow g \odot y$
8:         $ygt\_spec(ii, :) \leftarrow$ Absolute value of FFT-shifted FFT($yg$)
9:     end for
10:    Plot spectrogram using $ygt$ and the pcolor() command
11: end procedure
12: Repeat 4 to 11 for different $\sigma$        ▷ Window width
13: Repeat 4 to 11 for different $\Delta t$        ▷ Window translation (controls undersampling and oversampling)
14: Repeat 4 to 11 with different filter kernels
```

3.1.1 Data setup

The sound clip with a segment of Handel's "Messiah" is available in Matlab as the dataset `handel`. The dataset is loaded to the workspace and it contains the time-series pressure data and the sampling frequency, Fs, at which it is recorded. The time corresponding to k_{th} element in the data can be obtained by the ratio of the position k to the sampling frequency. The time domain is thus defined by dividing the positions of the datapoints with Fs. The frequency space is defined in $[-\frac{Fs}{2}, \frac{Fs}{2}]$, and the the Fourier domain has a discretization size of Fs/L, where L is the length of the signal. Visualizing the data in Fourier space allows us to see all the frequencies in the signal contained within $[-\frac{Fs}{2}, \frac{Fs}{2}]$, but does not provide the temporal information needed to reproduce the time-series data.

3.1.2 Gabor transform and visualization

Gabor transform is used here to get the time-dependent frequency domain information. This is done by first designing a filter kernel, such as a Gaussian kernel, centered at a time t_0 in the signal and with a width determined by its variance σ^2. FFT of the product of the kernel with the time-series data provides the frequency information contained in a window around t_0. By "sliding" the window by changing t_0 in increments of time, Δt, the frequency contents at different time points in the signal are obtained. This data is then plotted as a spectrogram to visualize the frequency information across time.

3.1.3 Effect of changing window width

Since the window width determines the range of frequencies that can be detected with an FFT, the data is analyzed with a range of window widths while holding the translation parameter, Δt, a constant. The results are visualized as spectrograms.

3.1.4 Undersampling and oversampling

The translation parameter Δt determines how quickly the window moves across the time domain. This also means that Δt determines the level of overlap, or lack thereof, between successive windows. By adopting a range of Δt for a given window width parameter σ, these differences are visualized.

3.1.5 Filter kernels

The filters used to define the windows where FFT is performed have a significant effect on the frequencies detected in the signal. Therefore, a few different filter kernels are used for the same translation Δt and comparable window widths. While the window width for Gaussian kernel can be adjusted by modifying σ suitably, the other filters' widths are set by manually adjusting their widths to match that of each other.

3.2 Piano and Recorder

Algorithm 2 provides a high-level technique to study the music notes from two recordings of *Mary had a little lamb*, one played with a piano and the other with a recorder.

Algorithm 2 Algorithm to differentiate the music generated by a piano and a recorder.

```
1: Import data
2: Define time and Fourier domains, visualize in both domains
3: Identify the first harmonic in piano and recorder data
4: Create Shannon windows around first harmonics for piano and recorder data
5: Using procedure from Algorithm 1 (lines 4 to 11), generate spectrograms for unfiltered and filtered data
6: Match the frequency-sequence in time to get the music score
```

3.2.1 Data setup

The recordings are loaded into the workspace, with their accompanying sampling frequencies. Similar to the method described in Section 3.1.1, the time domains and frequency domains are setup separately for the two recordings. The first harmonics are identified in each of the signals manually, noting that they are typically the frequencies with high amplitudes.

3.2.2 Filtering the data to generate the music score

Once the first harmonic is identified, the overtones can be filtered out to obtain a "clean" score. This is implemented with a step-function (Shannon) filter around the first harmonics, which acts as a bandpass filter. Spectrograms are then generated with the unfiltered and filtered signals for both the piano and the recorder. The music score can be easily identified by matching the frequency composition in time with the frequency designations for keys in a piano.

4 Computational Results

4.1 Spectrographic analysis of Handel's "Messiah"

Appendix B.1 shows the MATLAB code developed from the implementation of Algorithm 1. The data from the Matlab file `handel` is loaded into the workspace, and visualized in time and frequency domains. Throughout the implementation in Matlab, the fft-shifting of the frequency domain vectors are tracked and appropriately processed in order to preserve their order. Fig. 2 shows that while the time-series data contains a number of frequencies pertaining to the male and female voices, a violin, and other instruments, it is impossible to tell their temporal location from the FFT. In order to overcome this shortcoming, windowed Fourier transform is used with a Gaussian kernel serving as the window of width $\sigma = 0.0366$ seconds in the time domain, and a translation (Δt) of 0.061 seconds between successive windows. The result from this Gabor transformation is stored and plotted as a spectrogram using the `pcolor()` command, as can be seen in Fig. 3. The spectrogram and the FFT's are from real-valued functions; therefore, only the single-sided spectrum is shown and discussed in this section. Since the intensity of a point (t, ν) in the spectrogram is proportional to its amplitude, we can see that the frequencies corresponding to men (135–415 Hz) and women (250–980 Hz) singing, and the various other instruments at different frequencies. In the time domain, when there are "pauses" in the song, we can see dark vertical patches corresponding to the interval when the music pauses.

To explore the effect of changing the kernel window width, the translation parameter Δt is held constant at 0.0183 seconds while σ is set to the values $[0.0061, 0.0244, 0.1221, 0.6104]$ in the time domain. The spectrograms thus generated are plotted in Fig. 4. It can be seen that at lower values of σ, the resolution in the frequency domain is quite poor, especially at lower frequencies. This is because the filter window is quite narrow and cannot adequately track higher wavelength information. As the width increases, the frequency domain resolution improves. However, having very wide windows, like in Fig. 4)(c) and (d), results in a poorer resolution in the time domain while having very high resolution in frequency domain.

Similarly, changing Δt to [0.0024, 0.0610, 0.2441, 1.2207] seconds, while holding the window width parameter σ to 0.0244, we can observe how the amount of resampling affects the results, as seen in Fig. 5. When Δt is low,

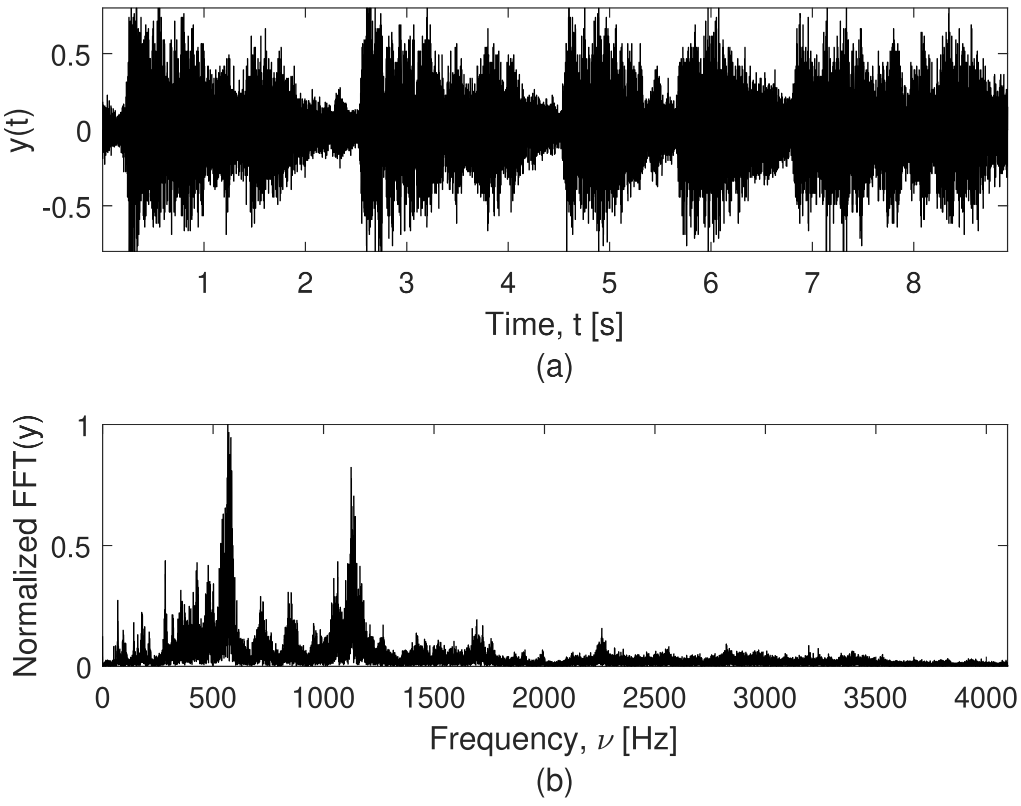

Figure 2: (a) The time trace of the pressure signal pertaining to the music clip; (b) FFT of the entire music clip.

there is significant overlap between successive windows, resulting in the accumulation of a minimally variant frequency information at short time intervals. This results in a spectrogram with good representation in both time and frequency domains, as in Fig. 5(a), and represents the oversampling of data. This could also result in a "grainy" or "sharp" spectrogram due to the high information density. On the other hand, at very large values of Δt, not enough frequency domain information is available at short time intervals to generate a clear spectrogram, and instead results in a "fuzzy" graph, such as Fig. 5(d). This other extreme is a case of undersampling, where the fuzziness results from the interpolation of frequency domain data between large time intervals.

Finally, we explore the effects of changing the filter kernels for the Gabor windows. Gaussian kernel, Mexican Hat kernel, Shannon (step function) window, and a custom triangular window, are implemented with the same translation in the time domain of $\Delta t = 0.0610$ seconds. The width of the windows are set to be roughly equivalent to that of the Gaussian kernel with $\sigma = 0.0244$. The results are shown in Fig. 6. With the way these filters are setup in this instance, we can see that the Gaussian and triangular filters offer the best resolution in both time and frequency domains, while the Mexican hat wavelet and the Shannon windows result in fuzzy spectrograms. This illustrates the fact that the particular choice of Gabor window and its construction can have a large impact on extracting useful information from a given data.

4.2 Music score

The MATLAB code to analyze the music files from the piano and recorder is given in Appendix B.2. After loading the datasets to the workspace, the time-domain and frequency domain information is visualized as shown in Fig. 7. It is readily observed in the FFT data in Fig. 7(b), which is normalized by the respective maxima, that both the piano and the recorder have the first harmonics in narrow ranges. A threshold of 0.25 in the normalized FFT is used to identify the frequencies of interest manually, and a step-function bandpass filter is constructed around them for the two signals. The filter is then multiplied elementwise with the FFT data to filter out the all other frequencies, and the time-domain signal is reconstructed from the filtered data. Gabor windows are then used to analyze the unfiltered and filtered data and construct spectrograms, as shown in Fig. 8.

It can be seen from Fig. 8(a) and (b) that both the piano and the recorder have multiple overtones of the fundamental frequencies. These are multiple strong overtones for the piano, especially in the frequency range below 4 kHz, while the recorder sound has less overtones. This contributes to the "richness" of sound associated with a

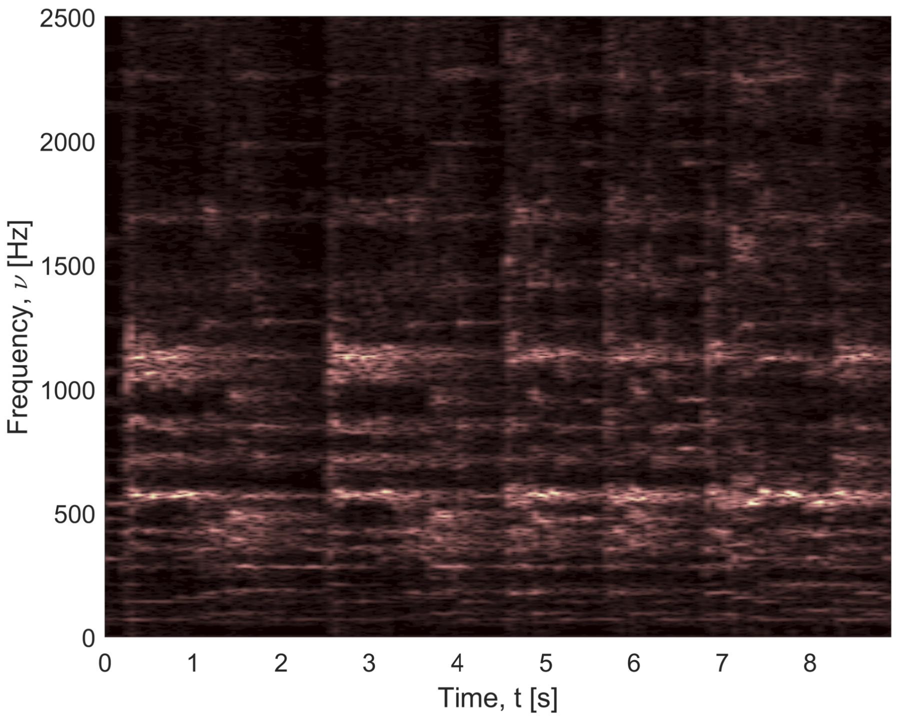

Figure 3: Spectrogram of the music clip produced with a Gaussian kernel ($\Delta t = 0.061$ seconds, $\sigma = 0.0366$), with the intensity in the graph proportional to the amplitude of frequency at that time.

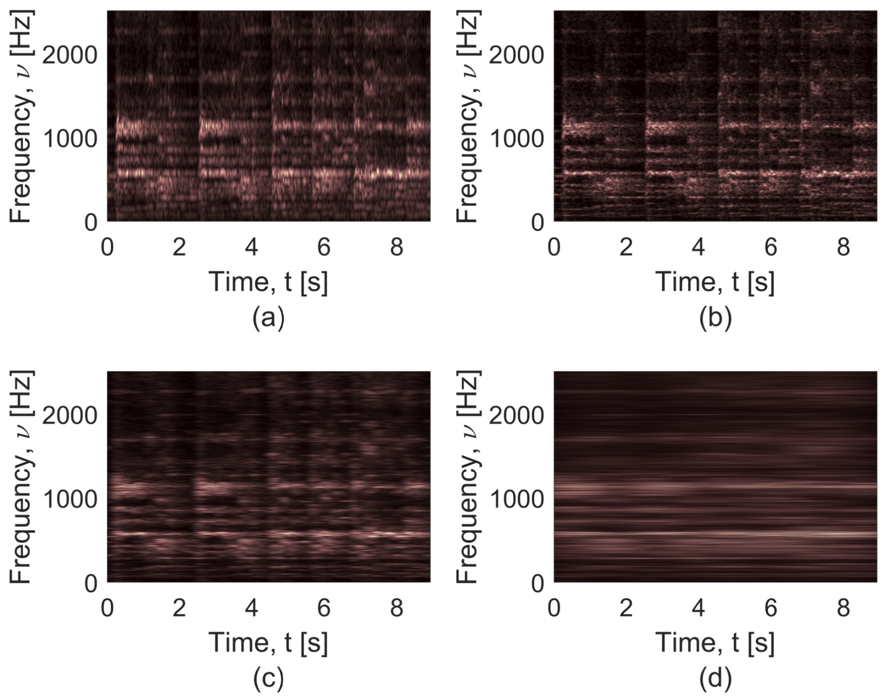

Figure 4: Spectrograms with a Gaussian kernel ($\Delta t = 0.0183$ seconds), with width of (a) $\sigma = 0.0061$; (b) $\sigma = 0.0244$; (c) $\sigma = 0.1221$; and (d) $\sigma = 0.6104$.

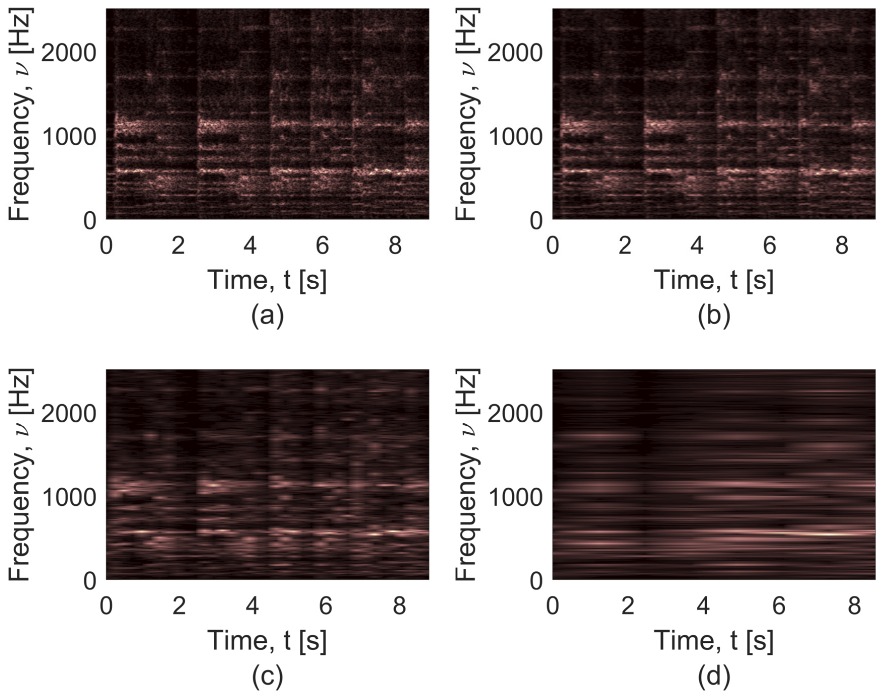

Figure 5: Spectrograms with a Gaussian kernel ($\sigma = 0.0244$), with translation parameter (a) $\Delta t = 0.0024$ seconds; (b) $\Delta t = 0.0610 seconds$; (c) $\Delta t = 0.2441$ seconds; and (d) $\Delta t = 1.2207$ seconds.

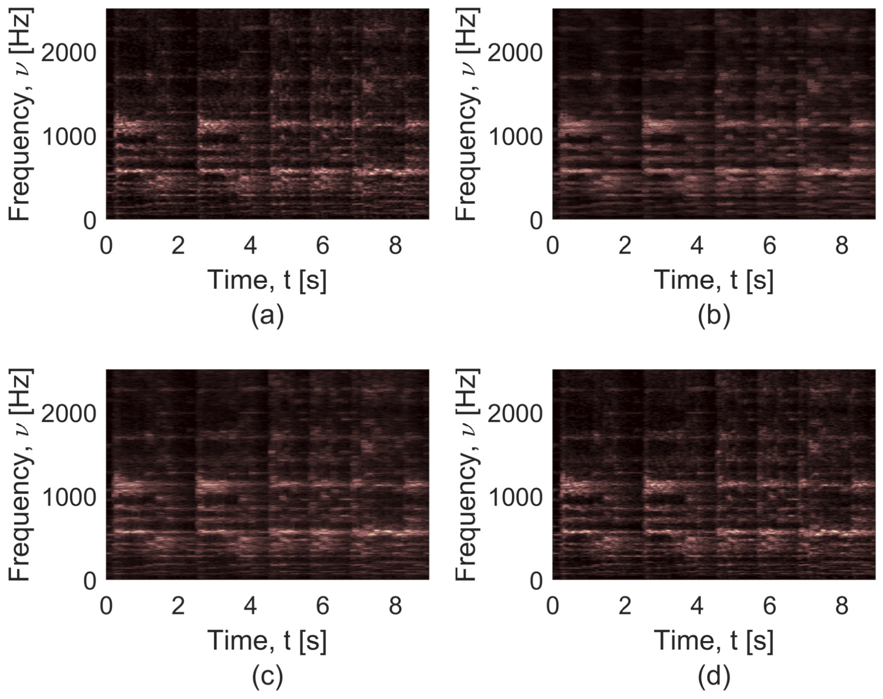

Figure 6: Spectrograms constructed with following kernels (with $\Delta t = 0.0610$ seconds): (a) Gaussian, $\sigma = 0.0244$; (b) Ricker, $\sigma = 0.0244$; (c) Shannon, $a = 0.0793$; and (d) Triangular, $a = 0.0793$.

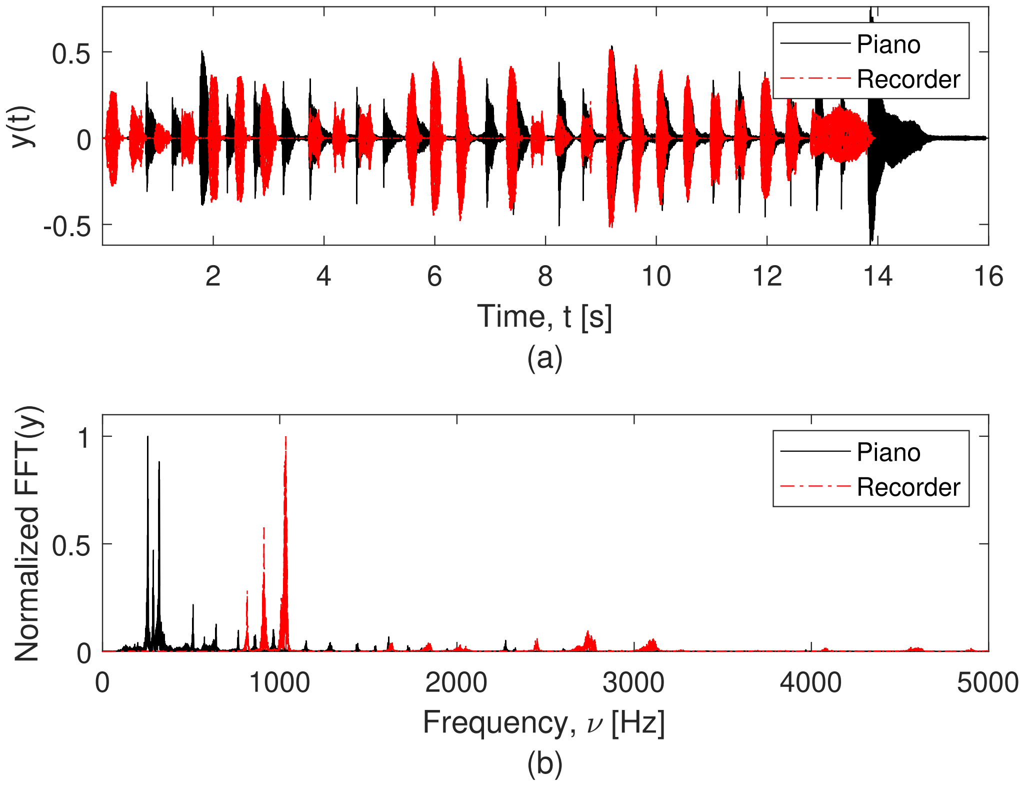

Figure 7: (a) Time-series pressure data of music recorded with a piano and a recorder; (b) FFTs of the recordings.

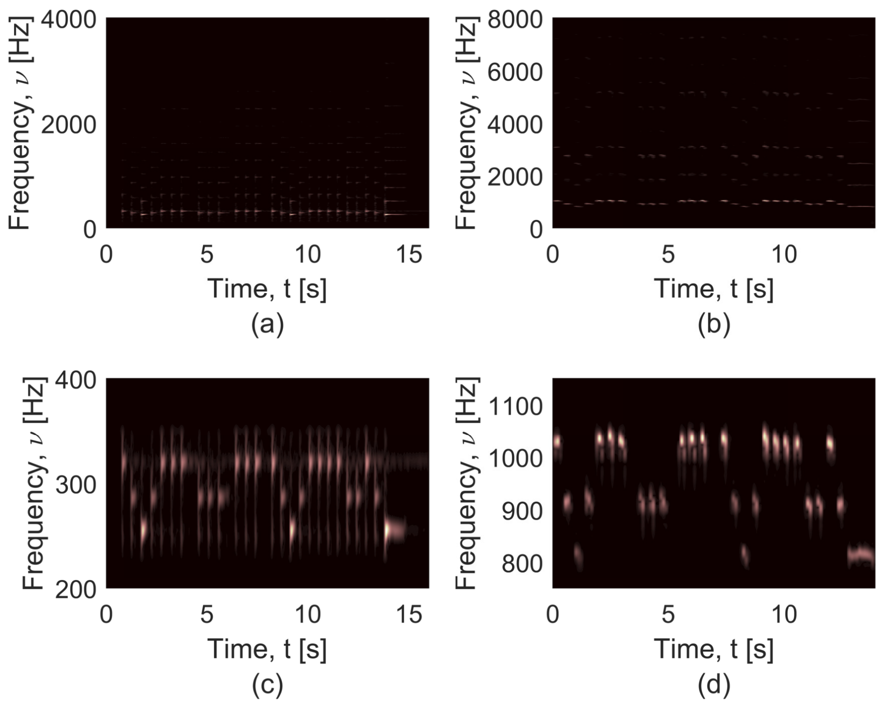

Figure 8: Spectrograms of recordings of (a) piano, unfiltered; (b) recorder, unfiltered; (c) piano, with overtones filtered out; (d) recorder, with overtones filtered out.

piano. Fig. 8(c) and (d) show the spectrograms after the overtones are filtered out for the piano and the recorder data respectively. The recorder music is ”cleaner” whereas the piano music displays sounds in a continuous range of frequencies between the fundamental harmonics. The frequencies are matched with the piano keys, and it is quickly realized that the recorded music is in scientific pitch [2, 3]. After making the necessary corrections to match the frequencies in a typical scale with C_4 at 261.62 Hz [4, 5], the music scores are generated and presented in Table 1.

Instrument	Fundamental frequencies [Hz]	Music score in scientific pitch
Piano	255.7, 287.1, 319.8	$E_4D_4C_4D_4E_4E_4E_4D_4D_4D_4E_4E_4E_4$
		$E_4D_4C_4D_4E_4E_4E_4E_4D_4D_4E_4D_4C_4$
Recorder	816.7, 910.9, 1034	$C_6A^\#_5G^\#_5A^\#_5C_6C_6C_6A^\#_5A^\#_5A^\#_5C_6C_6C_6$
		$C_6A^\#_5G^\#_5A^\#_5C_6C_6C_6C_6A^\#_5A^\#_5C_6A^\#_5G^\#_5$

Table 1: Computational results.

5 Summary and Conclusions

This work demonstrates the power of windowed Fourier transforms to provide frequency information in terms of time. Using clips from Handel's ”Messiah”, we see that the construction and implementation of Gabor filter has a significant impact on the information extracted from the data. Similarly, working with the music recordings from the piano and the recorder shows the power of spectral analysis in filtering out noise and reconstructing ”clean” music score. Thus the Gabor transform is a method that builds upon the strength of FFT to help glean time-dependent frequency domain information.

References

[1] Steven L. Brunton and J. Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control.* Cambridge University Press, 2019. DOI: `10.1017/9781108380690`.

[2] *Frequencies for equal-tempered scale, A_4 = 440 Hz.* URL: `https://pages.mtu.edu/~suits/notefreqs.html`. (accessed: 02/14/2020).

[3] *Scientific pitch.* URL: `https://en.wikipedia.org/wiki/Scientific_pitch`. (accessed: 02/14/2020).

[4] Alexander J. Ellis and Alfred J. Hipkins. “Tonometrical Observations on Some Existing Non-Harmonic Musical Scales”. In: *Proceedings of the Royal Society of London* 37 (1884), pp. 368–385. ISSN: 03701662. URL: `http://www.jstor.org/stable/114325`.

[5] *Cent(music).* URL: `https://en.wikipedia.org/wiki/Cent%5C_(music)`. (accessed: 02/14/2020).

Appendix A MATLAB Functions

Here is an overview of the major functions used in the programs in Appendix B.

- `pcolor(X, Y, C)` creates a 2-D plot on a grid formed by vertices (`X`$_i$`, Y`$_i$) from the vectors `X` and `Y`. The matrix C determines the color of the cell which has a vertex (`X`$_k$`, Y`$_k$), where `k` is the first of the four vertices of the cell encountered.
- `shading interp` specifies that the surface plot will have interpolated values between its faces or lines.
- `Y = fft(X)` returns the FFT of the vector `X`, or if `X` is a matrix, the FFTs of the columns of `X`. `Y` is of the same size as `X`.
- `Y = ifft(X)` returns the discrete inverse Fourier transform of the vector `X`, or if `X` is a matrix, then the discrete inverse Fourier transform of the columns of `X`.
- `Y = fftshift(X)` rearranges Fourier transform `X` so that the zero frequency component is at the center of the array. `X` can be a vector, a matrix, or higher-dimensional array.
- `X = ifftshift(Y)` performs the inverse of `fftshift()` and rearranges `Y` so that the frequency components are in the same order as the original FFT output.

Appendix B MATLAB Code

B.1 Analysis of Handel's "Messiah"

This is the script used to analyze Handel's "Messiah" and identify the spectral composition over time.

```
clear; close all; clc

%% Load data

load handel
v = y' / 2;

L = length(v);
t = (1:L)';
figure, plot(t / Fs, v);
xlabel('Time [sec]')
ylabel('Amplitude')
title('Signal of Interest, v(n)')

freq_unshift = Fs * (0: (L - 1)) / L;
freq_unshift(((L + 1) / 2 + 1):end) =...
    freq_unshift(((L + 1) / 2 + 1):end) - Fs;   % Frequencies for FFT
freq = fftshift(freq_unshift);
yt = fft(y);
Y = abs(yt) * 2 / L;    % Single-sided spectrum
Y(1) = Y(1) / 2;    % DC adjustment

fig1 = figure;
fig1.Units = 'inches';
fig1.Position = [-.1 1.8 6.75 5.0625];
fig1.PaperUnits = 'inches';
fig1.PaperSize = [6.75 5.0625];
s1 = subplot(2,1,1); % Time domain
plot(t / Fs, y, 'k')
set(gca,'Fontsize', 12), xlabel({'Time, t [s]', '(a)'}), ylabel('y(t)')
axis('tight')

s2 = subplot(2,1,2); % Fourier domain
plot(freq, fftshift(Y) / max(Y), 'k');
set(gca, 'Fontsize', 12)
xlabel({'Frequency, \nu [Hz]', '(b)'}), ylabel('Normalized FFT(y)')
axis([0 (Fs / 2) 0 1])

% print('handel', '-depsc', '-r600')
% print('handel', '-dpng', '-r600')

%%  GABOR TRANSFORM - ILLUSTRATION

shannon_filt = zeros(length(t), 1);
[g_filt, g_filt_trans, g_filt_width, shannon_filt(4e4 - 650 : 4e4 + 650)] =...
    deal(exp(-(t - 4e4) .^ 2 / (2 * 400 ^ 2)), exp(-(t - 4.3e4) .^ 2 /...
    (2 * 400 ^ 2)), exp(-(t - 4e4) .^ 2 / (2 * 1000 ^ 2)), 1);  % Different kernels
filters = [g_filt, g_filt_trans, g_filt_width, shannon_filt];
y_filt = filters .* y;
y_filt_t = fft(y_filt);

fig2 = figure(2);
```

```
fig2.Units = 'inches';
fig2.Position = [0 4 6.75 5.0625];
fig2.PaperUnits = 'inches';
fig2.PaperSize = [6.75 5.0625];
s1 = subplot(3, 1, 1);
h1 = plot(t / Fs, y, 'k');
hold on
h2 = plot(t / Fs, g_filt, 'r');
h3 = plot(t / Fs, g_filt_trans, 'm-.');
h4 = plot(t / Fs, g_filt_width, 'g');
h5 = plot(t / Fs, shannon_filt, 'b');
ylabel('y(t), f(t)')
xlabel({'Time, t [s]', '(a)'})
xlim([4.3 5.8])
legend([h1], {'Original signal'})

s2 = subplot(3, 1, 2);
s2.Box = 'on';
hold on
h6 = plot(t / Fs, y_filt(:, 1), 'r');
h7 = plot(t / Fs, y_filt(:, 2), 'm-.');
h8 = plot(t / Fs, y_filt(:, 3), 'g');
h9 = plot(t / Fs, y_filt(:, 4), 'b');
ylabel('y(t)f(t)')
xlabel({'Time, t [s]', '(b)'})
xlim([4.3 5.8])
legend('Gaussian', 'Gaussian, shifted', 'Gaussian, wider', 'Shannon')

s3 = subplot(3,1,3);
s3.Box = 'on';
hold on
h10 = plot(freq, abs(fftshift(y_filt_t(:, 1))) / max(abs(y_filt_t(:, 1))), 'r');
h11 = plot(freq, abs(fftshift(y_filt_t(:, 2))) / max(abs(y_filt_t(:, 2))), 'm-.');
h12 = plot(freq, abs(fftshift(y_filt_t(:, 3))) / max(abs(y_filt_t(:, 3))), 'g');
h13 = plot(freq, abs(fftshift(y_filt_t(:, 4))) / max(abs(y_filt_t(:, 4))), 'b');
xlim([0 1200])
ylim([0 1])
ylabel('FFT(yf)/max(FFT(yf))')
xlabel({'Frequency, \nu [Hz]', '(c)'})

s1.FontSize = 12; s2.FontSize = 12; s3.FontSize = 12;

% print('Gabor_illustration', '-depsc', '-r600')
% print('Gabor_illustration', '-dpng', '-r600')

%% SPECTROGRAM

tslide = 0:500:L;
ygt_spec = nan(length(tslide), L);
for ii = 1:length(tslide)
    g_filt = exp(-(t - tslide(ii)) .^ 2 / (2 * 300 ^ 2)); % Gaussian Gabor kernel
    yg = g_filt .* y;
    ygt = fft(yg);
    ygt_spec(ii, :) = abs(fftshift(ygt'));
%     subplot(3,1,1), plot(t, y, 'k', t, g_filt, 'r')
%     subplot(3,1,2), plot(t, yg, 'k')
%     subplot(3,1,3), plot(freq, abs(fftshift(ygt)) / max(abs(ygt)))
%     axis([0 2500 0 1])
%     drawnow
%     pause(0.1)
end

fig3 = figure(3);
pcolor(tslide / Fs, freq, ygt_spec.'),
shading interp
ylim([0 2500])
xlabel('Time, t [s]')
ylabel('Frequency, \nu [Hz]')
% set(gca, 'Ylim', [-50 50], 'Fontsize', [14])
colormap(pink)
fig3.Units = 'inches';
fig3.Position = [0 4 6.75 5.0625];
fig3.PaperUnits = 'inches';
fig3.PaperSize = [6.75 5.0625];
ax = gca;
ax.FontSize = 12;
% print('Spectrogram', '-depsc', '-r600')
% print('Spectrogram', '-dpng', '-r600')

%% EFFECT OF CHANGING WINDOW WIDTH

fig4 = figure(4);
labels = {'(a)', '(b)', '(c)', '(d)'};
sig = [50, 200, 1000, 5000];    % Gaussian kernel width parameter
for jj = 1: 1: length(sig)

    tslide = 0:150:L;
    ygt_spec = nan(length(tslide), L);
    for ii = 1:length(tslide)
        g_filt = exp(-(t - tslide(ii)) .^ 2 / (2 * sig(jj) ^ 2));
        yg = g_filt .* y;
        ygt = fft(yg);
        ygt_spec(ii, :) = abs(fftshift(ygt'));
    end

    subplot(2, 2, jj)
    pcolor(tslide / Fs, freq, ygt_spec.'),
    shading interp
    ylim([0 2500])
    xlabel({'Time, t [s]', labels{jj}})
    ylabel('Frequency, \nu [Hz]')
    set(gca, 'Fontsize', 12)
    colormap(pink)
end
fig4.Units = 'inches';
fig4.Position = [0 4 6.75 5.0625];
fig4.PaperUnits = 'inches';
fig4.PaperSize = [6.75 5.0625];
% print('Spectrogram_width', '-depsc', '-r600')
% print('Spectrogram_width', '-dpng', '-r600')

%% EFFECT OF CHANGING TRANSLATION PARAMETER

fig5 = figure(5);
```

```
slide = [20; 500; 2000; 10000]; % Translation parameter
for jj = 1: 1: length(slide)

    tslide = 0:slide(jj):L;
    ygt_spec = nan(length(tslide), L);
    for ii = 1:length(tslide)
        g_filt = exp(-(t - tslide(ii)) .^ 2 / (2 * 200 ^ 2));
        yg = g_filt .* y;
        ygt = fft(yg);
        ygt_spec(ii, :) = abs(fftshift(ygt'));
    end

    subplot(2, 2, jj)
    pcolor(tslide / Fs, freq, ygt_spec.'),
    shading interp
    ylim([0 2500])
    xlabel({'Time, t [s]', labels{jj}})
    ylabel('Frequency, \nu [Hz]')
    set(gca, 'Fontsize', 12)
    colormap(pink)
end

fig5.Units = 'inches';
fig5.Position = [0 4 6.75 5.0625];
fig5.PaperUnits = 'inches';
fig5.PaperSize = [6.75 5.0625];
% print('Spectrogram_shifted', '-depsc', '-r600')
% print('Spectrogram_shifted', '-dpng', '-r600')

%% EFFECT OF DIFFERENT FILTERS

tslide = 0:500:L;
ygt_spec = nan(length(tslide), L);
ymht_spec = nan(length(tslide), L);
ysht_spec = nan(length(tslide), L);
ytrt_spec = nan(length(tslide), L);

for ii = 1:length(tslide)

    g_filt = exp(-(t - tslide(ii)) .^ 2 / (2 * 200 ^ 2)); % Gaussian filter
    mh_filt = 2 / (sqrt(3 * 200) * pi ^ .25) * ...
            (1 - ((t - tslide(ii)) ./ 200) .^ 2) ...
            .* exp(- (t - tslide(ii)) .^ 2 ./ (2 * 200) ^ 2);
    mh_filt = mh_filt / max(mh_filt);   % Mexican hat filter
    shannon_filt = zeros(length(t), 1); % Shannon filter
    triangle_filt = zeros(length(t), 1);    % Triangular filter
    if tslide(ii) < 650
        shannon_filt(1:tslide(ii) + 650) = 1;
        triangle_filt(1:tslide(ii) + 650) = ...
            interp1([tslide(ii) - 650, tslide(ii), tslide(ii) + 650],...
            [0, 1, 0], 1:tslide(ii) + 650);
    elseif tslide(ii) > (L - 650)
        shannon_filt(tslide(ii) - 650:L) = 1;
        triangle_filt(tslide(ii) - 650:L) = ...
            interp1([tslide(ii) - 650, tslide(ii), tslide(ii) + 650],...
            [0, 1, 0], tslide(ii) - 650:L);
    else
```

```
        shannon_filt(tslide(ii) - 650 : tslide(ii) + 650) = 1;
        triangle_filt(tslide(ii) - 650 : tslide(ii) + 650) = ...
            interp1([tslide(ii) - 650, tslide(ii), tslide(ii) + 650],...
            [0, 1, 0], t(tslide(ii) - 650 : tslide(ii) + 650));
    end

    yg = g_filt .* y;
    ygt = fft(yg);
    ygt_spec(ii, :) = abs(fftshift(ygt'));

    ymh = mh_filt .* y;
    ymht = fft(ymh);
    ymht_spec(ii, :) = abs(fftshift(ymht'));

    ysh = shannon_filt .* y;
    ysht = fft(ysh);
    ysht_spec(ii, :) = abs(fftshift(ysht'));

    ytr = triangle_filt .* y;
    ytrt = fft(ytr);
    ytrt_spec(ii, :) = abs(fftshift(ytrt'));
end

fig6 = figure(6);
s1 = subplot(2, 2, 1);
pcolor(tslide / Fs, freq, ygt_spec.'),
shading interp
ylim([0 2500])
xlabel({'Time, t [s]', '(a)'})
ylabel('Frequency, \nu [Hz]')
colormap(pink)
s2 = subplot(2, 2, 2);
pcolor(tslide / Fs, freq, ymht_spec.'),
shading interp
ylim([0 2500])
xlabel({'Time, t [s]', '(b)'})
ylabel('Frequency, \nu [Hz]')
colormap(pink)
s3 = subplot(2, 2, 3);
pcolor(tslide / Fs, freq, ysht_spec.'),
shading interp
ylim([0 2500])
xlabel({'Time, t [s]', '(c)'})
ylabel('Frequency, \nu [Hz]')
colormap(pink)
s4 = subplot(2, 2, 4);
pcolor(tslide / Fs, freq, ytrt_spec.'),
shading interp
ylim([0 2500])
xlabel({'Time, t [s]', '(d)'})
ylabel('Frequency, \nu [Hz]')
colormap(pink)
s1.FontSize = 12; s2.FontSize = 12; s3.FontSize = 12; s4.FontSize = 12;
fig6.Units = 'inches';
fig6.Position = [0 4 6.75 5.0625];
fig6.PaperUnits = 'inches';
fig6.PaperSize = [6.75 5.0625];
```

```
% print('Spectrogram_filters', '-depsc', '-r600')
% print('Spectrogram_filters', '-dpng', '-r600')
```

B.2 Analysis of sounds from a piano and a recorder

This is the script used to study the differences in compositions of music from a piano and a recorder.

```
clear; close all; clc

%% LOADING AND DEFINING DATASETS

tr_piano = 16;  % recording time in seconds
y_piano = audioread('music1.wav');
L_piano = length(y_piano);
t_piano = (1:L_piano)';
Fs_piano = L_piano / tr_piano;
% p8 = audioplayer(y_piano, Fs_piano); playblocking(p8);

tr_rec = 14;  % recording time in seconds
y_rec = audioread('music2.wav');
L_rec = length(y_rec);
t_rec = (1:L_rec)';
Fs_rec = L_rec / tr_rec;
% p8 = audioplayer(y_rec, Fs_rec); playblocking(p8);

%% FFTs

freq_piano = Fs_piano * ((-L_piano / 2):1:(L_piano / 2 - 1)) / L_piano;
freq_rec = Fs_rec * ((-L_rec / 2):1:(L_rec / 2 - 1)) / L_rec;
y_pianot = fft(y_piano);
y_rect = fft(y_rec);
Y_piano = abs(y_pianot) * 2 / L_piano;
Y_rec = abs(y_rect) * 2 / L_rec;
Y_piano(1) = Y_piano(1) / 2;
Y_rec(1) = Y_rec(1) / 2;
% psd_piano = (1 / (Fs_piano * L_piano)) * abs(Y_piano) .^ 2;
% psd_rec = (1 / (Fs_rec * L_rec)) * abs(Y_rec) .^ 2;

fig1 = figure;
fig1.Units = 'inches';
fig1.Position = [-.1 1.8 6.75 5.0625];
fig1.PaperUnits = 'inches';
fig1.PaperSize = [6.75 5.0625];
s1 = subplot(2,1,1); % Time domain
plot(t_piano / Fs_piano, y_piano, 'k')
hold on
plot(t_rec / Fs_rec, y_rec, 'r-.');
set(gca,'Fontsize', 12), xlabel({'Time, t [s]', '(a)'}), ylabel('y(t)')
axis('tight')
legend('Piano', 'Recorder')

s2 = subplot(2,1,2); % Fourier domain
plot(freq_piano, fftshift(Y_piano) / max(Y_piano), 'k');
hold on
plot(freq_rec, fftshift(Y_rec) / max(Y_rec), 'r-.');
set(gca, 'Fontsize', 12)
xlabel({'Frequency, \nu [Hz]', '(b)'}), ylabel('Normalized FFT(y)')
ylim([0 1.1])
```

```matlab
% xlim([0, max([Fs_piano / 2, Fs_rec / 2])])
xlim([0 5e3])
legend('Piano', 'Recorder')

% print('mhall', '-depsc', '-r600')
% print('mhall', '-dpng', '-r600')

%% SPECTROGRAM WITH GABOR WINDOW AND FILTERING

tslide_piano = 0:1000:L_piano;
tslide_rec = 0:5e3:L_rec;

% Shannon windows
filt_piano = zeros(length(Y_piano), 1);
filt_piano(abs(freq_piano) > 230 & abs(freq_piano) < 350) = 1;
filt_rec = zeros(length(Y_rec), 1);
filt_rec(abs(freq_rec) > 785 & abs(freq_rec) < 1065) = 1;

y_pianoft = y_pianot .* ifftshift(filt_piano);
y_pianof = ifft(y_pianoft);
y_recft = y_rect .* ifftshift(filt_rec);
y_recf = ifft(y_recft);

% Unfiltered Piano Spectrogram
ygt_spec_piano = nan(length(tslide_piano), L_piano);
for ii = 1:length(tslide_piano)
    g_filt_piano = exp(-(t_piano - tslide_piano(ii)) .^ 2 / (2 * 1500 ^ 2)); % Gabor
    yg_piano = g_filt_piano .* y_piano;
    ygt_piano = fft(yg_piano);
    ygt_spec_piano(ii, :) = abs(fftshift(ygt_piano'));
end

fig = figure;
fig.Units = 'inches';
fig.Position = [-.1 1.8 6.75 5.0625];
fig.PaperUnits = 'inches';
fig.PaperSize = [6.75 5.0625];
s1 = subplot(2, 2, 1);
pcolor(tslide_piano / Fs_piano, freq_piano, ygt_spec_piano.'),
shading interp
% ylim([0 2500])
xlabel({'Time, t [s]', '(a)'})
ylabel('Frequency, \nu [Hz]')
ylim([0 4000])
s1.FontSize = 12;
colormap(pink)
clearvars ygt_spec_piano

% Filtered Piano spectrogram
ygt_spec_pianof = nan(length(tslide_piano), L_piano);
for ii = 1:length(tslide_piano)
    g_filt_piano = exp(-(t_piano - tslide_piano(ii)) .^ 2 / (2 * 1500 ^ 2)); % Gabor
    yg_pianof = g_filt_piano .* y_pianof;
    ygt_pianof = fft(yg_pianof);
    ygt_spec_pianof(ii, :) = abs(fftshift(ygt_pianof'));
end
```

```
s3 = subplot(2, 2, 3);
pcolor(tslide_piano / Fs_piano, freq_piano, ygt_spec_pianof.'),
shading interp
% ylim([0 2500])
xlabel({'Time, t [s]', '(c)'})
ylabel('Frequency, \nu [Hz]')
ylim([200 400])
s3.FontSize = 12;
colormap(pink)
clearvars ygt_spec_pianof

% Unfiltered Recorder spectrogram
ygt_spec_rec = nan(length(tslide_rec), L_rec);
for ii = 1:length(tslide_rec)
    g_filt_rec = exp(-(t_rec - tslide_rec(ii)) .^ 2 / (2 * 1200 ^ 2)); % Gabor
    yg_rec = g_filt_rec .* y_rec;
    ygt_rec = fft(yg_rec);
    ygt_spec_rec(ii, :) = abs(fftshift(ygt_rec'));
end

s2 = subplot(2, 2, 2);
pcolor(tslide_rec / Fs_rec, freq_rec, ygt_spec_rec.'),
shading interp
% ylim([0 2500])
xlabel({'Time, t [s]', '(b)'})
ylabel('Frequency, \nu [Hz]')
ylim([0 8e3])
s2.FontSize = 12;
colormap(pink)
clearvars ygt_spec_rec

% Filter Recorder spectrogram

ygt_spec_recf = nan(length(tslide_rec), L_rec);
for ii = 1:length(tslide_rec)
    g_filt_rec = exp(-(t_rec - tslide_rec(ii)) .^ 2 / (2 * 1200 ^ 2)); % Gabor
    yg_recf = g_filt_rec .* y_recf;
    ygt_recf = fft(yg_recf);
    ygt_spec_recf(ii, :) = abs(fftshift(ygt_recf'));
end

s4 = subplot(2, 2, 4);
pcolor(tslide_rec / Fs_rec, freq_rec, ygt_spec_recf.'),
shading interp
% ylim([0 2500])
xlabel({'Time, t [s]', '(d)'})
ylabel('Frequency, \nu [Hz]')
s4.YLim = [750 1150];
s4.FontSize = 12;
colormap(pink)
% print('mhall_sp_analysis', '-depsc', '-r600')
% print('mhall_sp_analysis', '-dpng', '-r600')

clearvars ygt_spec_recf
```