Homework 2: Gábor Transforms

Joshua Yap
AMATH 482
Winter 2020

February 8, 2020

Abstract

This paper explores the Gábor transform and applies it to sound recordings. We are able to extract a music score from the recordings because of time-frequency analysis enabled by the Gábor transform.

1 Introduction and Overview

This homework is split into 2 parts: Part I requires us to analyze Handel's Messiah using time-frequency analysis; Part II requires us to reproduce a music score for *Mary had a Little Lamb* by filtering two recordings. Gábor transforms are used in both parts to give us frequency information of the signal at different times.

2 Theoretical Background

The Gábor transform addresses a key limitation of the Fourier transform: it tells us what frequencies are present in the signal but fails to locate the points in time that these frequencies occur. For our application of analyzing a song, the Fourier transform is not very useful, ie. it does not tell us the times at which each note is played. The Gábor transform gives us this information by defining a time window over which the signal is filtered. Mathematically, it is given by

$$\tilde{f}(\tau,\omega) = \int_{-\infty}^{\infty} f(t)g(t-\tau)e^{-i\omega t}dt, \tag{1}$$

where $f(t)$ is the given signal and $g(t-\tau)$ is the Gábor filter. The choice of filter shape is up to the user and we implement a Gaussian filter for our application, such that

$$g(t-\tau) = e^{-a(t-\tau)^2}. \tag{2}$$

As in the last homework, parameter a determines the filter width (a higher a gives us a narrower filter) and parameter τ determines where the filter is centered. We slide this filter down the time axis to pick out the

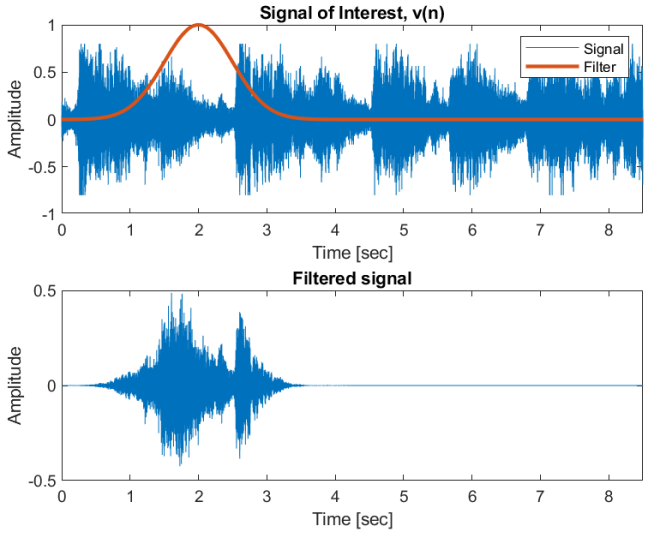


Figure 1: The upper plot shows the original signal with the overlay of a Gábor filter, the bottom plot shows the resulting signal once the filter is applied. The parameter selections for this filter are $a = 2$ and $\tau = 2$.

range of frequencies at different times. Figure 1 is an illustration of how the filter is applied to a signal.

The Gábor transform exhibits properties of Heisenberg's uncertainty principle: a narrow filter gives us more time information and less frequency information, while a wide filter gives us more frequency information but less time information. In time-frequency analysis, we are interested in tuning our filter to give us the best balance of time and frequency information for our application.

3 Algorithm Implementation and Development

3.1 Part I

The goal of this part was to explore a portion of Handel's Messiah using time-frequency analysis. I chose to construct a Gábor transform and explore how changing parameters of the transform would affect the output, a spectrogram. A spectrogram is a diagram that conveys time and frequency information in the same plot. The following outlines how the spectrogram was generated.

Define domains: Domains were defined in time and frequency. Since the algorithm uses the Fast Fourier transform, each domain requires 2^n points. However, since the data was from a real recording, it did not have exactly 2^n points, so I appended zeros to the end of the vector containing the signal to round it up to the nearest 2^n. This is known as zero-padding.

Apply Gábor transform: A Gaussian filter was constructed at t = 0 using Equation 2. It was then applied to the original signal to produce a filtered signal.

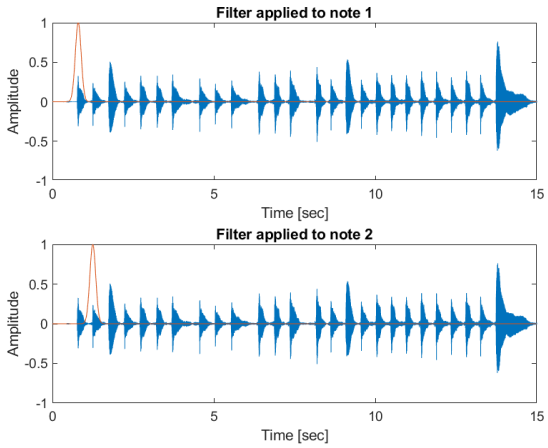


Figure 2: The upper plot shows the filter applied to the first pulse and the lower plot shows the filter applied to the second pulse.

Slide filter: The filter was translated along the time axis in small steps by incrementing the Each time it was translated, the filter was applied to the signal to produce a new filter signal. Each of these filtered signals was stored as a row in a matrix.

Generate spectrogram: The matrix was plotted as a color plot with colors representing different amplitudes.

3.2 Part II

The task in this part was to load two recordings of *Mary had a Little Lamb*, one from a piano and one from a recorder, and identify the notes that are played and the times at which the notes are played. The Gábor transform was again used to extract this information. Outlined below is the process.

Define domains: Time and frequency domains were defined and the signal was zero-padded.

Locate pulses: Each note produced a signal pulse on the graph, so these pulses were located so that the filter could be centered on them.

Apply Gábor transform: The transform was applied to each pulse to identify the frequencies contained in them. Figure 2 illustrates the transform applied to the first two pulses.

Determine frequencies: Each pulse contained a base frequency ω_0 as well as overtones $2\omega_0$, $3\omega_0$, etc. The base frequency was extracted since it is the one that determines the note being played. The overtones give each instrument its distinctive sound. The frequencies extracted were then converted to Hertz by $f = \omega/2\pi$ and looked up in a table to see what notes they correspond to.

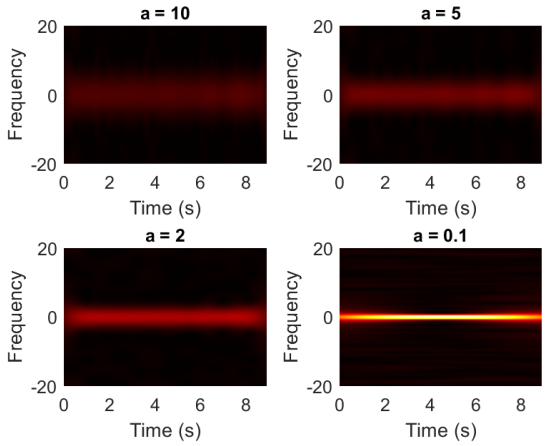


Figure 3: Spectrograms of Handel's Messiah for different filter widths. A narrow filter gives better time resolution while a wide filter gives better frequency resolution.

4 Computational Results

4.1 Part I

After producing one spectrogram of the signal, I explored how different filter widths would affect it. As can be seen from Figure 3, a narrower filter (corresponding to a larger a) produces a fainter spectrogram more spread out in frequency, while a wider filter produces one that is bright but contains mostly low frequencies. This happens because a narrow filter acts like a Dirac delta-function, picking out the value of the frequency at a point. It does not, however, encapsulate a large number of frequencies. A wide filter encapsulates more frequencies, as well as greater amplitudes of frequencies, which is why the spectrogram is brighter.

I also explored oversampling and undersampling by varying the distance we move the filter for each step. An oversampled signal would be one where we have a lot of overlap when moving the filter. An undersampled

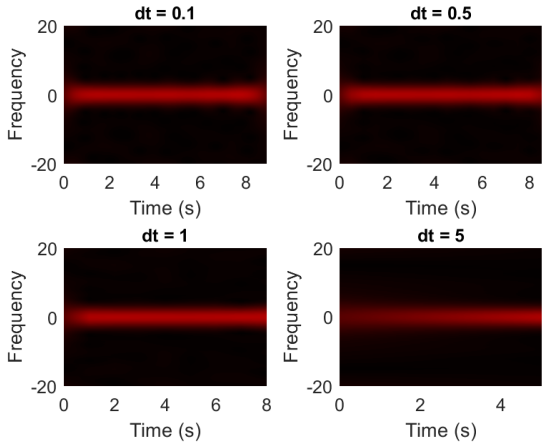
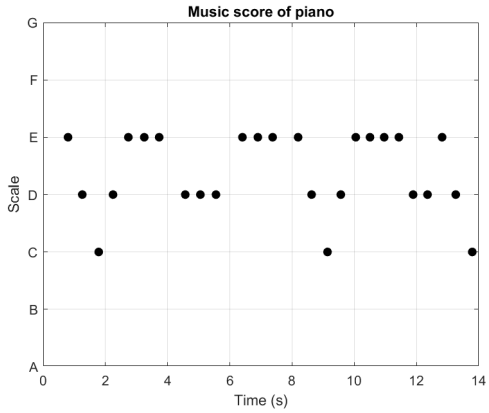
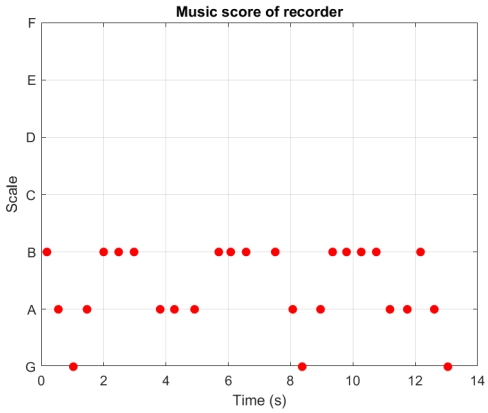


Figure 4: Spectrograms for different values of dt plotted for a filter with $a = 2$.



(a) Music score for the piano. Note C on this scale is middle C. Each dot represents the note being played at the corresponding time.



(b) Music score for the recorder. This scale is 2 octaves higher than the piano scale. Each dot represents the note being played at the corresponding time.

Figure 5: Music scores obtained from frequency information from the recordings.

signal is one where we have little to no overlap of the filter. Figure 4 shows the spectrograms for different step lengths dt. At first glance there does not appear to be much difference. Undersampled signals (large dt) get cut off at the end and appear slightly dimmer, but they exhibit the same properties.

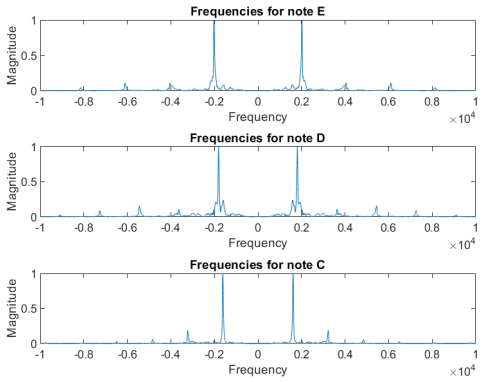
4.2 Part II

Using the algorithm described in the previous section, we obtained the music scores for both instruments as illustrated in Figure 5. As expected, the piano had notes with lower frequencies, but both scores exhibit a similar note pattern.

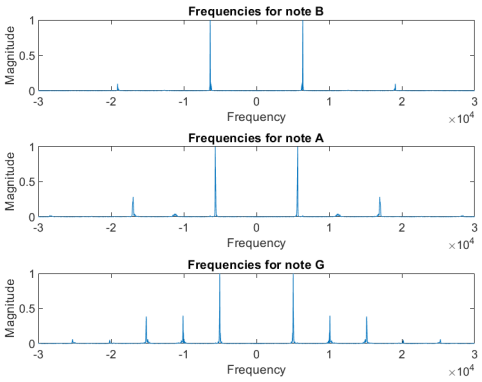
We can also look at the frequency plots of both instruments to see what overtones are present. Figure 6 compares frequencies from the piano and the recorder. The center frequencies of each note are the spikes with lowest frequency. The smaller peaks away from the center are the overtones. Both instruments have overtones at integer multiples of the center frequency which causes those of the recorder to be more spread out. Both sets of plots have a similar pattern, but the plot for the recorder spans a greater range of frequencies, with its center frequencies higher than the center frequencies of the piano. This is what makes the recorder notes two octaves higher than the piano.

5 Summary and Conclusions

Part I required us to gain competency in using the Gábor transform, which helped when applying the transform in Part II. We were able to find out the notes played on both instruments and identified some overtones that make the instruments sound different.



(a) Frequencies of notes from piano.



(b) Frequencies of notes from recorder.

Figure 6: Center frequency with overtones represented by peaks at integer multiples of center frequency.

Appendices

Appendix A

Summary of MATLAB functions used

fft: Transforms original signal from 3-space to frequency space.

fftshift: Shifts values of Fourier transformed signal to match basis frequencies.

max: Finds the maximum value in array.

pcolor: Plots spectrogram using colored surface.

audioread: Converts sound file to vector in MATLAB.

Appendix B

MATLAB code

```
close all; clear; clc
load handel
v = y';
t = (1:length(v))/Fs;
subplot(2,1,1)
plot(t,v)
xlim([0 8.5])
```

```
hold on
xlabel('Time [sec]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');

% Define domains
n = pow2(nextpow2(length(v))); % 2^17
L = n/Fs;
t2 = [0:1/Fs:L];
t = t2(1:n);
k = (2*pi/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);

% Define Gabor filter
a = 2; % controls filter width
tau = 2; % center of filter
filter = exp(-a*(t-tau).^2);
plot(t,filter,'Linewidth',2)
hold off
legend('Signal','Filter')

% Apply Gabor filter to signal
v_pad = [v zeros(1,n-length(v))]; % zero-padding v
v_filt = filter.*v_pad;
v_filt_t = fft(v_filt);

% Spectrograms for varying window sizes
figure()
a_vec = [10 5 2 0.1];
for jj = 1:length(a_vec)
    a = a_vec(jj);
    tslide = [0:0.1:t(length(v))];
    vft_spec = zeros(length(tslide),n);
```

```
    for ii = 1:length(tslide)
        g = exp(-a*(t-tslide(ii)).^2);
        vf = g.*v_pad;
        vft = fft(vf);
        vft_spec(ii,:) = fftshift(abs(vft));
    end
    subplot(2,2,jj)
    pcolor(tslide,ks,vft_spec.'), shading interp
    set(gca, 'Ylim',[-20,20],'Fontsize',12)
    title(['a = ',num2str(a)],'Fontsize',12)
    xlabel('Time (s)')
    ylabel('Frequency')
    colormap(hot)
    caxis([0 100])
end

%% Explore over and undersampling
dt_vec = [0.1 0.5 1 5];
for jj = 1:length(dt_vec)
    a = 2;
    dt = dt_vec(jj);
    tslide = [0:dt:t(length(v))];
    vft_spec = zeros(length(tslide),n);
    for ii = 1:length(tslide)
        g = exp(-a*(t-tslide(ii)).^2);
        vf = g.*v_pad;
        vft = fft(vf);
        vft_spec(ii,:) = fftshift(abs(vft));
    end
    subplot(2,2,jj)
    pcolor(tslide,ks,vft_spec.'), shading interp
    set(gca, 'Ylim',[-20,20],'Fontsize',12)
    title(['dt = ',num2str(dt)],'Fontsize',12)
```

```
    xlabel('Time (s)')
    ylabel('Frequency')
    colormap(hot)
    caxis([0 100])
end

%% Part 2
% Piano
[y,Fs] = audioread('music1.wav');
tr_piano=length(y)/Fs; % record time in seconds
% plot((1:length(y))/Fs,y);
% xlabel('Time [sec]'); ylabel('Amplitude');
% title('Mary had a little lamb (piano)');
% p8 = audioplayer(y,Fs); playblocking(p8);

a = 60;
mindist = 0.2;
minheight = 0.2;
[f_piano,t_piano,yft_piano] = get_frequencies(y',Fs,a,mindist,minheight,false);

%%
% Recorder
figure(2)
[y,Fs] = audioread('music2.wav');
tr_rec=length(y)/Fs; % record time in seconds

figure()
plot((1:length(y))/Fs,y);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (recorder)');
% p8 = audioplayer(y,Fs); playblocking(p8);
[f_rec,t_rec,yft_rec] = get_frequencies(y',Fs,60,0.3,0.08,false);
```

```
end


function [f_max,tpulse,yft] = get_frequencies(y,Fs,a,minpeakdist,minpeakheight,flag)
% Takes a vector y of signal data and sampling frequency Fs and
% returns the center frequencies in the signal

% Define domains
n = pow2(nextpow2(length(y))); % 2^20
L = n/Fs;
t2 = [0:1/Fs:L];
t = t2(1:n);
k = (2*pi/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);

y_pad = [y zeros(1,n-length(y))]; % zero-padding y

% Find location of pulses
[pks,tpulse] = findpeaks(y_pad,Fs,'MinPeakDistance',minpeakdist,'MinPeakHeight',minpeakheight);
yft = zeros(length(tpulse),n);
for ii = 1:length(tpulse)
    g = exp(-a*(t-tpulse(ii)).^2);
    yf = g.*y_pad;
    yft(ii,:) = fft(yf);
end

% Find maxima of frequency data
[M,I] = max(abs(yft),[],2);
omega = k(I);
f_max = omega/(2*pi);

end
```