AMATH 482 Winter 2020

Homework 2:Gábor Transforms

Tanner Graves

Introduction and Overview

Abstract

The following is an exploration of methods used in signal analysis, specifically, audio processing. As frequency analysis is an indispensable tool, we will make extensive use of the Gábor Transform to produce the spectrograms necessary for analyzing our signal. From there we will be able to filter and then utilize the time-frequency information extract musical information(timbre of an instrument, as well as note pitch, time and duration).

Part 1: Handel

To get used to implementing the Gábor Transform, we will utilizes Handel's Messiah as a ginnie pig. In computing our Gábor Transform, we are faced with several decisions along the way. Including: which filter do we use to create our window, how wide should that window be, and how many windows should we use to extract either the time data or the frequency data that we desire. A naive approach might, to try keep increasing our window numbers to give more time information and, at the same time, keep increasing our window width to get more frequency information. As we shall see, worries of undersampling, oversampling, and an unavoidable trade off between time and frequency information will result in a more complicated situation.

Part 2: Mary had a Little Lamb

Equipped with the knowledge of how to implement and use the Gábor Transform, we can now demonstrate some of its uses for time-frequency analysis. A spectrogram represents different frequencies and when they are present in a signal. We could, however, instead think about this musically instead of a signal. Sheet music that a musician would read primarily communicates what notes are played and when. I will demonstrate the information available after creating a spectrogram may readily be converted into sheet music with a recording of ‘Mary had a Little Lamb’.

With this information there is additionally a lot to be learned about the characteristics of a particular sound or instrument. If a guitar and a piano are both to play the same note, we can recognize it as the same note, yet there is some disparity in information that accounts for this difference. This difference can roughly be accounted for by timbre, or the overtones present in a sound. Overtones are simply integer multiples of the base frequency that is the ‘note’. Using the Gábor Transform, we can see these overtones and observe how they vary across two instruments playing the same note. Here, a piano and recorder both playing ‘Mary had a Little Lamb’.

Theoretical Background

The Gábor Transform is a similar tool and even bears a strong relation to the Fourier Transform. They both simularally are able to extract frequency data from a signal. However, fourier has the notable issue that all time information is lost in the transform. Meaning it may indicate that a frequency is present in a signal, but there is no way of telling where in time it happened. This is the main problem the Gábor Transform seeks to address. Instead of concerning ourselves with all the time information at once, the gabor transform works by filtering a signal into different windows which may be analyzed one at a time.

This filter must have the property that it approaches 0 as t goes to ± ∞, making a gaussian a natural choice. However, several other filters could be considered depending on application with some notable features as being not symmetric or oscillating wildly. Once a filter is chosen, the formula for the Gábor Transform is precisely the Fourier transform of the filter shifted by a parameter τ multiplied by the signal, meaning this will now be a function of two parameters: frequency and time.

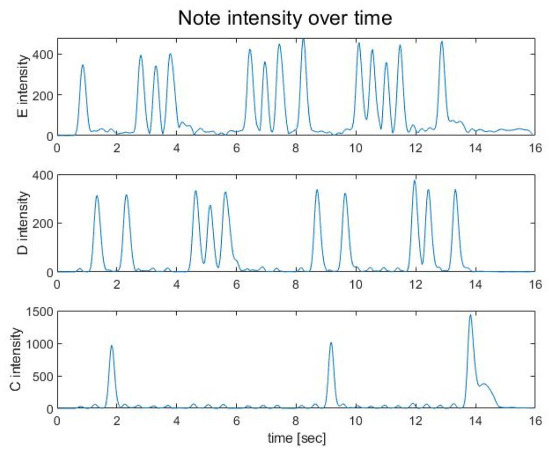
$$G[f](t,\omega) = \int_{-\infty}^{\infty} f(\tau)g^*(\tau - t)e^{-i\omega\tau}\,d\tau$$

Note this is a complex value function, and here * denotes the complex conjugate. Notable properties of the Gabor Transform include: linearity and invertibility. One notable limitation of the gabor transform is a necessary trade off between time and frequency information. This is due to the variance of the time and frequencies being related. What this practically means, is that if we aim to make our window incredibly small, as to get more time data, we will be losing frequency as well as vice-versa.

Algorithm Implementation and Development

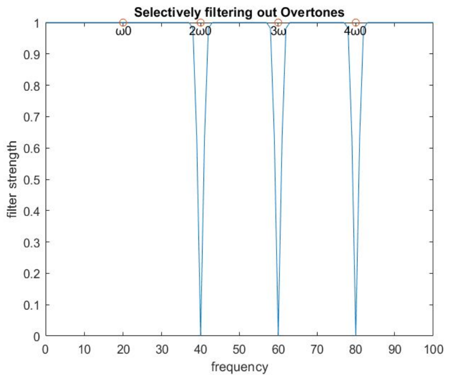
The most technical aspect of this project was creating my own implementation of the Gábor Transform. Everything else was largely a matter of application. At the end of the code in appendix B is a custom function to compute the Gábor Transform given the signal, sample frequency, window width, and number of windows. This function will return a matrix of the frequency intensities as well as the times and frequencies that correspond to the entries in the matrix. Lecture code comprises most of the code with accommodations made for the possibility that the passed signal could have either an even or odd number of entries.

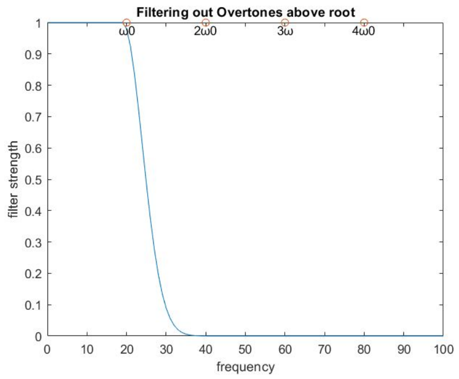
Converting the spectrogram data into a musical score would have been an interesting optimization problem - trying to find all the peaks and seeing what frequencies and times they correspond to, but there were a few factors that made a more discritized approach more appropriate. Managing the multiple optima that correspond to multiple frequencies seemed a little unusual, additionally, since we're asking which notes are being played, instead of which frequencies, it made more sense to take slices of the spectrogram at each note. At each of the frequencies that correspond to a note that could be in our song we may keep track of the intensity of that note over time. The intensity being over a certain threshold would indicate the note currently being played, and under would indicate not being played at a particular time. For 'Mary had a Little Lamb' played on the piano we get the following data:



Observing the peaks along each plot provides a clear indication of when a note is active. Detecting note start and duration is a simple matter of detecting the positive derivative when the threshold is crossed that corresponds to a note starting and the negative when the note ends.

A robust system meant to convert a music signal into written music would have to account for not just three notes as in this example, but the rest of the entire piano roll, and possibly all the notes comfortably within the range of human hearing. With these additional notes comes additional considerations; overtones will likely register as played notes. A clear response to this would be to filter the overtones out. In an application where we would want to detect multiple notes playing at once, it would make sense to use notch (gaussian) filters to take out the overtones specifically. In this application we don't want anything above the root, so we can just use a cutoff filter.

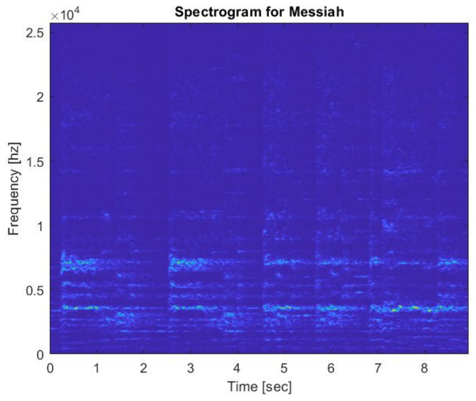




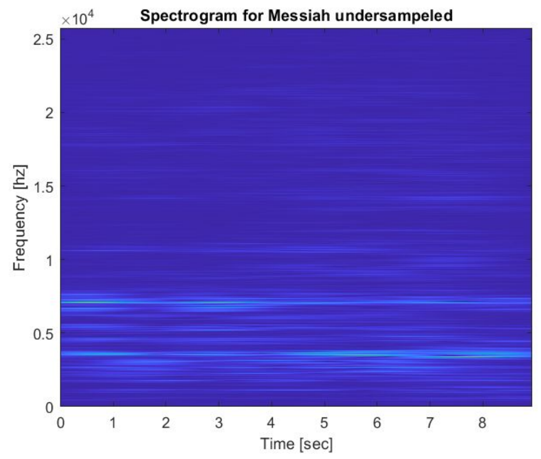
Computational Results

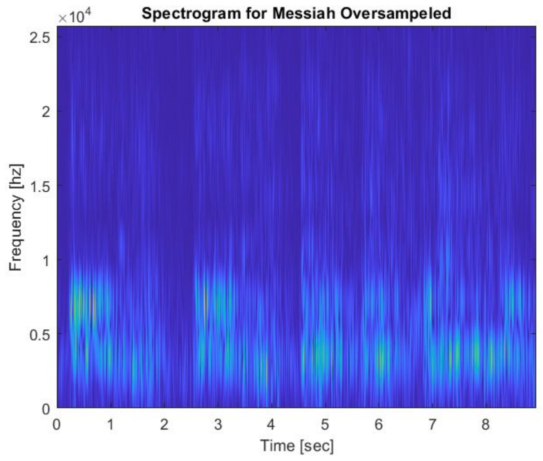
Part 1: Handel

The recording of Handel's Messiah provided an opportunity to play around with implementing and applying these tools. Below is a spectrogram for the recording. Immediately what's striking is the amount going on. Listening to the song, there aren't many instruments playing, the most prominent sound, if not the only, is voice. That said, in comparison to the simpler samples we will analyze later, there's quite a lot of information portrayed on the spectrogram. This is to speak primarily to the more complicated timbre of the human voice when compared to instruments like piano or recorder. The spectrogram also reveals some more odd quirks about the human voice, like concinates like the 'L' in "hallelujah" doesn't come through as strongly as the vowels. Their presence in the spectrogram is less in both intensity and concentration, that is to say it looks kinda fuzzy. Indicative of many different frequencies composing the sound.



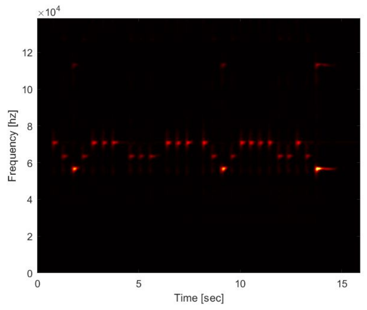
A large part of this assignment was tweaking the window parameters for my transform. Initially, I struggled with accidently undersampling my data. I didn't intuit how quickly the song was actually sampled. This resulted in a lot of streaky horizontal lines as can be seen in the left figure. However, as we keep refining our window further and further, we approach taking the transform of frequencies, which are not just well outside the range of human hearing and can't really be interpreted as sound, but nearing the limit of frequencies that can be represented at our sample rate. Adjusting this value, however, made me realize that working with audio is quite nice; there is a large range of window sizes that provide acceptable results, and the 'Heisenberg uncertainty principle' type effect is something that we need not really grapple with on the scale we experience sound.

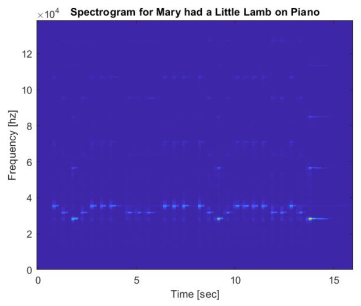


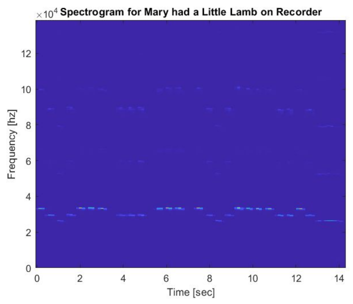


Part 2: Mary had a Little Lamb

The musical information was extracted from the piano recording. See appendix B for a table of note played, start time, and duration. The notes can be nicely visualized in time by viewing the spectrogram (left figure below).







We may also observe the differences in timbre using the spectrogram of the piano (center figure above) and the recorder (right figure). Where the piano has overtones steadily decaying, the tone of the recorder seems much more 'pure', that is, the overall intensity of the overtones is low, not that the instrument is actually pleasant. Additionally, the recorder has a much stronger second overtone (4ω) than first (2ω).

Summary and conclusion

Gabor Transform has proved an effective tool for audio analysis that requires both time and frequency information. Used both as a visual tool for interpretation, or for processing and filtering, as we demonstrated with three separate audio files: Messiah and Mary had a Little Lamb played on both piano and recorder.

Appendix A. MATLAB Functions

```
imagesc(x,y,S) - a useful tool for visualizing data that displays the
values of a given matrix scaled as a picture
flipud(A) - useful command that will flip a matrix along its rows
sgtitle('str') - plot titles that support formatting and subplots
```

Appendix B. MATLAB Code and Data

```
clear; close all; clc
%% Handel
figure(1)
load handel;
plot((1:length(y))/Fs,y);
n = length(y);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');
%p8 = audioplayer(y,Fs); playblocking(p8);
[spec,t,ks] = myGabor(y',Fs,1000,750);
spec_zoom = spec(n/2+0.5:end,:);
k_zoom = ks(n/2+0.5:end);
ylim([0 0.35*10^4])
figure(10)
imagesc(t,flipud(k_zoom),spec_zoom);
title('Spectrogram for Messiah');
xlabel('Time [sec]');
ylabel('Frequency [hz]');
set(gca,'YDir','normal');
%% Music 1
clear all;
figure(3)
[y,Fs] = audioread('music1.wav');
n = length(y);
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);
%figure(4)
[spec,t,ks] = myGabor(y',Fs,100,100);
spec_zoom = spec(n/2+1:n/2+1+2*10^4,:);
k_zoom = ks(n/2+1:end);
ylim([0 0.35*10^4])
figure(10)
imagesc(t,flipud(k_zoom),spec_zoom);
title('Spectrogram for Mary had a Little
Lamb on Piano');
xlabel('Time [sec]');
ylabel('Frequency [hz]');
%colormap(hot);
set(gca,'YDir','normal');
E = spec_zoom(5150,:);
D = spec_zoom(4590,:);
C = spec_zoom(4075,:);
tp = linspace(0,tr_piano,length(E));
E_notes = note_finder(E,tp,100);
D_notes = note_finder(D,tp,100);
C_notes = note_finder(C,tp,100);
E_notes_dur =
[E_notes(1,:);E_notes(2,:)-E_notes(1,:)];
D_notes_dur =
[D_notes(1,:);D_notes(2,:)-D_notes(1,:)];
C_notes_dur =
[C_notes(1,:);C_notes(2,:)-C_notes(1,:)];

figure(11);
subplot(3,1,1);
plot(tp,E);
ylabel('E intensity');
subplot(3,1,2);
plot(tp,D);
ylabel('D intensity');
subplot(3,1,3);
plot(tp,C);
ylabel('C intensity');
xlabel('time [sec]')
sgtitle('Note intensity over time');
%% Music 2
clear all;
figure(5)
[y,Fs] = audioread('music2.wav');
n = length(y);
tr_rec=length(y)/Fs; % record time in
seconds
plot((1:length(y))/Fs,y);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (recorder)');
%p8 = audioplayer(y,Fs); playblocking(p8);
figure(6)
[spec,t,ks] = myGabor(y',Fs,100,100);

spec_zoom = spec(n/2+1:n/2+1+6*10^4,:);
k_zoom = ks(n/2+1:end);
ylim([-0.5*10^4 0.5*10^4])
figure(12)
imagesc(t,flipud(k_zoom),spec_zoom);
```

```
title('Spectrogram for Mary had a Little
Lamb on Recorder')
xlabel('Time [sec]');
ylabel('Frequency [hz]');
%colormap(hot);
set(gca,'YDir','normal');
%% Functions
function result = cutoff(k, k0)
A = zeros(1,length(k));
for i = 1:length(k)
    if i <= k0
        A(i) = 1;
    else
        %A(i) = max([0 1-0.1*(i-k0)]);
        A(i) = exp(-0.02*(i-k0).^2);
    end
    result = A;
end
end

function arr = note_finder(Sn, t,thresh)
last = 0;
notes = zeros(2,1);
for j = 2:length(Sn)
    if Sn(j) >= thresh
        if Sn(j-1) < thresh
            notes = [notes [t(j);0]];%mark
the start of a new note
        end
    elseif Sn(j) < thresh && notes(end,end)
== 0
        notes(end,end) = t(j);
    end
end
arr = notes(:,2:end);
end

function [spec,t,ks] =
myGabor(S,Fs,a,num_T)
n=length(S); L = n/Fs;
t2=linspace(0,L,n+1); t=t2(1:n);
if mod(n,2) == 1
    k=(2*pi/L)*[0:n/2-0.5 -n/2+0.5:-1];
else
    k=(2*pi/L)*[0:n/2-1 -n/2:-1];
end
ks=fftshift(k);
tslide=linspace(0,L,num_T);
Sgt_spec = zeros(length(tslide),n);
for j=1:length(tslide)
    g=exp(-a*(t-tslide(j)).^2);
    Sg=g.*S;
    Sgt=fft(Sg);
    Sgt_spec(j,:) = fftshift(abs(Sgt)); %
We don't want to scale it
end
spec = transpose(Sgt_spec);
end
```

Note	Time played (Sorted)	Duration
E	0.733127025	0.3187508804
D	1.211253346	0.3187508804
C	1.657504578	0.3825010565
D	2.199381075	0.3187508804
E	2.677507396	0.3187508804
E	3.187508804	0.2868757924
E	3.633760037	0.3825010565
D	4.526262502	0.2868757924
D	5.004388823	0.2868757924
D	5.514390232	0.3506259685
E	6.311267433	0.3506259685
E	6.85314393	0.2550007044
E	7.299395162	0.3825010565
E	8.096272363	0.3506259685
D	8.574398684	0.3187508804
C	9.020649917	0.3506259685
D	9.530651325	0.2868757924
E	9.94502747	0.3825010565
E	10.3912787	0.3825010565

E	10.90128011	0.2868757924
E	11.34753134	0.3187508804
D	11.82565766	0.3187508804
D	12.30378399	0.2868757924
E	12.71816013	0.3506259685
D	13.19628645	0.3187508804
C	13.64253768	1.051877905