AMATH 482 Homework 2

Section 1. Introduction and Overview

While performing Fourier Transform throughout the whole signal is useful to recover the frequency signatures, we lose information about when each of the characteristic frequencies occur, which is an important information. Therefore, we can use Gabor Filtering to recover characteristic frequencies that occurs in a time window throughout the whole signal. We can set the sampling rate and the time window in Gabor filtering to recover a spectogram which conveys the amplitude of the characteristic frequencies and when they occur.

The first purpose of this project is to understand the effect of choosing sampling rate and the time window to the resolution of the spectrogram recovered. This concept is done through analyzing a musical piece by Handel.

The second purpose is to understand the spectrogram recovered from various musical instruments. The spectrogram will convey the signature frequencies of each time slot with its overtones that dictates the musical instrument's timbre. By performing filtering to the fourier transform of each time frame, we can find the fundamental frequencies that allow us to reproduce the musical score from the signal.

Section 2. Theoretical Background

In addition to fourier transform concept and using gaussian filter in the frequency domain described in Homework 1, this section describes the new filters used in this assignment.

The new concept introduced in this project are Gabor Filters and various types of band pass filters. In this assignment, they are implemented to the time domain signal to only obtain frequency data from a specified time slice within the whole signal. The Fourier transform of the signal will then only show the frequencies that is contained in the current time slice. This concept is the backbone of performing Gabor Filtering by using wavelets.

The Gabor Filter most widely used in this project takes the form described in Equation 1.

$$g(t) = \exp[(twindow * (t - tslide))^{10}]$$

Equation 1. Super Gaussian Filter

Where **twindow** corresponds to the width of the time window, and **tslide** corresponds to the center of the filter. This allows exclusion of data points outside the time window. In addition, the edges of this filter is a smooth steep curve, which will prevent the frequencies recovered to have high frequencies since the edges are smoothed out (eliminates discontinuity). Figure 1 visualizes the shape of this super Gaussian filter.

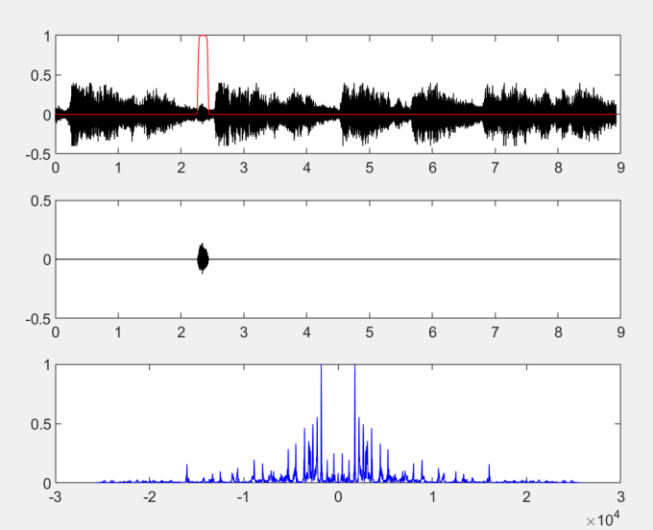


Figure 1. Super Gaussian Filter

The next filter used in this assignment is the Mexican Hat Filter, Equation 2 shows its equation.

$$g(t) = \frac{2}{\sqrt{3\sigma}\pi^{\frac{1}{4}}}(1 - (\frac{t - tslide}{sigma})^2))e^{-\frac{t-tslide}{2\sigma^2}}$$

Equation 2. Mexican Hat Filter

This filter has wiggles at its ends. The parameter **tslide** still corresponds to the center of the filter. Figure 2 visualizes the shape of this filter.

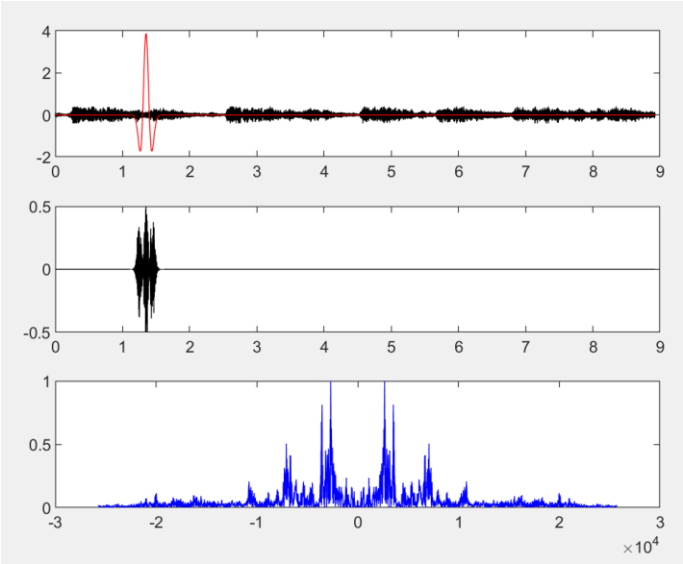


Figure 2. Mexican Hat Filter

Section 3. Algorithm Implementation and Development

To create a plot of the amplitude versus time of the music signal of interest, we first load a vector containing those amplitudes with an even length (this guarantees that the length of the wave number, k, to have the same length as the amplitude data). We then prepare the Fourier coefficients by loading the time vector which is just the length of the amplitude data all scaled to the sampling rate Fs. The length L pertains to the last element of time, and the total wave number n pertains to the length of the time vector. Lastly, since the Fourier Transform in Matlab is in the 2π domain, we scale the wave numbers to span L by multiplying each wave number with $2\pi/L$.

Next, we prepare the sampling rate and the time window used in the Gabor Filter. The sampling rate corresponds to the increment of time that spans from the beginning (t = 0s) until the end. The time window corresponds to the width of the time window used.

The Gabor Filtering is performed by using a for loop that takes each time slice, and performing fourier transform to the Gabor filtered signal that selects only frequencies within the time window, which is obtained by multiplying the filter with the signal. The resulting Gabor Filtered signal is then fourier transformed to obtain the frequencies occuring at that time window. The Fourier Transform of each time slice is then stored in a matrix, with each collumn pertains to the Fourier Transform of the time slice.

After all the Fourier Transform of the time slice has been recovered, we can plot the results in a spectrogram, with the X-axis being the time, Y-axis being the frequencies in Hz, and the color being the amplitude.

Furthermore, when analyzing the spectrogram from sound recordings of different instruments, we would need to filter out the overtones to recover the musical scores. This step is done by performing an additional Gaussian filtering to the Gabor Filtered signal of each time slot. I specified the tau (filter width) of the 2D Gaussian to be 0.001, and the Gaussian filter is implemented at the frequency domain. The center of this filter is determined by the frequency in each time slice which contains the strongest amplitude (the fundamental frequency). This is done through finding the index of the maximum amplitude, then substituding the index to the wave number to obtain the correct filter center. This filter is multiplied with the Fourier transform of the Gabor filtered signal to obtain the Gaussian filtered frequency domain signal of each time slot. Finally, we can take the inverse Fourier transform to obtain the time domain signal of the filtered out overtones. This time domain signal can be continuously added in each of the time slot to recover the full time domain signal to recover the sound recording which excludes the overtones. The logic behind this algorithm is to use different center frequency for the filter at each timeslot since we know that there are distinct frequencies occuring at each time slot, which corresponds to the musical pitch contained in the musical score.

After we successfully recover the spectrogram, we can check the frequencies of each characteristic frequency, which corresponds to its musical score. I manually looked up the characteristic frequencies and reproduced the musical instrument by looking up each characteristic frequencies.

Section 4. Computational Results

Figure 3. describes the spectrograms of Messiah by Handel generated with the Super Gaussian filter of different properties.

Firstly, the top left graph represents the base case where the sampling rate used is 0.05s and the twindow is 12s. The result shows all the frequencies available at each time step with relatively average resolution in both time and frequency. When the same sampling rate but higher twindow of 100s is used (top right), we can tell that the frequency resolution is higher. In other words, we can see more frequencies especially the higher ones since larger time windows mean that more frequencies are included. However, we sacrifice the clarity in time resolution as a consequence.

Next, we kept the twindow as default at 12s while severly reducing the tsample to 1s (bottom left). We can tell that the frequency resolution is much higher than the base case. However, we sacrifice time resolution since less time slices are observed. Lastly, the bottom right spectrogram is created by keeping the tsample at 0.05s while increasing the twindow to 100s. The resulting graph is similar to the bottom right graph because if the time window used is high, even though the sampling time is kept high, they all include similar frequency ranges which will lower the time resolution. This phenomenon is the result of oversampling.

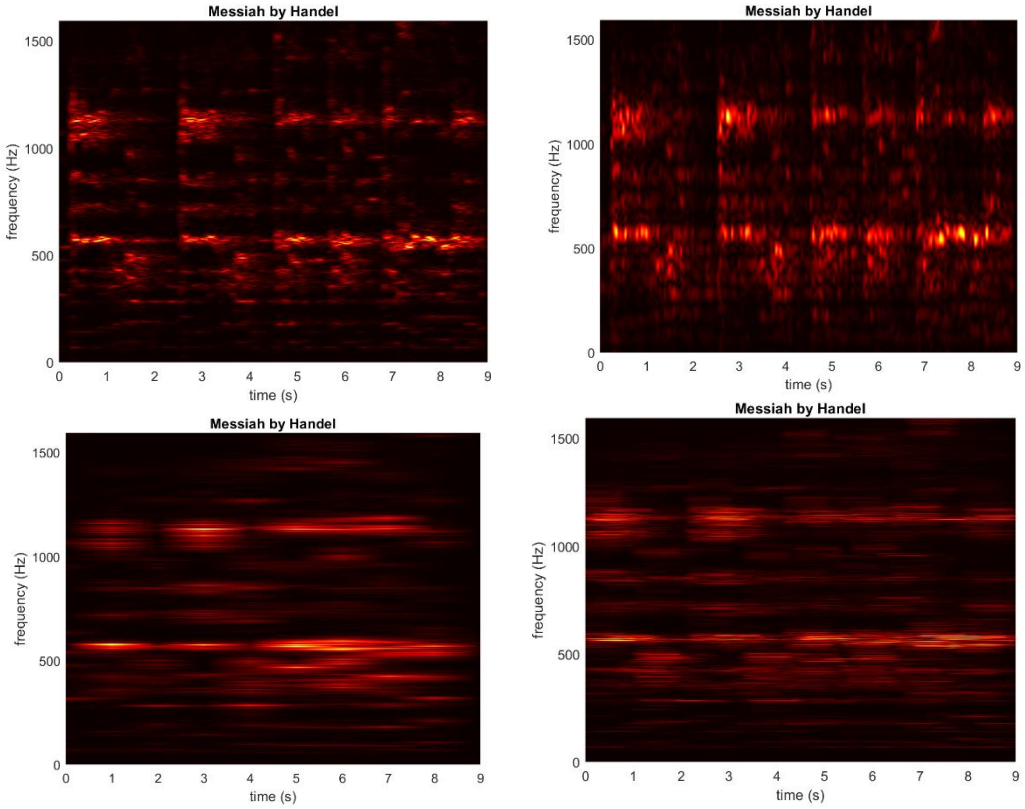


Figure 3. Spectrograms of Messiah by Handel

Top left: twindow = 12s, tsample = 0.05s || Top right: twindow = 100s, tsample = 0.05s

Bottom left: twindow = 12s, tsample = 1s || Bottom right: twindow = 2s, tsample = 0.05s

Figure 4 shows the spectrogram generated from “music1.wav”, which is the song Mary Had a Little Lamb played in piano using the technique described earlier. Observing the unfiltered spectrogram, we can see the overtones of each note which is represented by a faint spot on top of some of the pitch. In order to only analyze the fundamental frequencies only, eliminating the overtones, a gaussian filter is applied to each of the characteristic frequency. The resulting spectrogram is showed in the right graph in Figure 4. By looking at each of the pitch and using the Music Scores in Hz given, the complete music score is reproduced in the top graph of Figure 6.

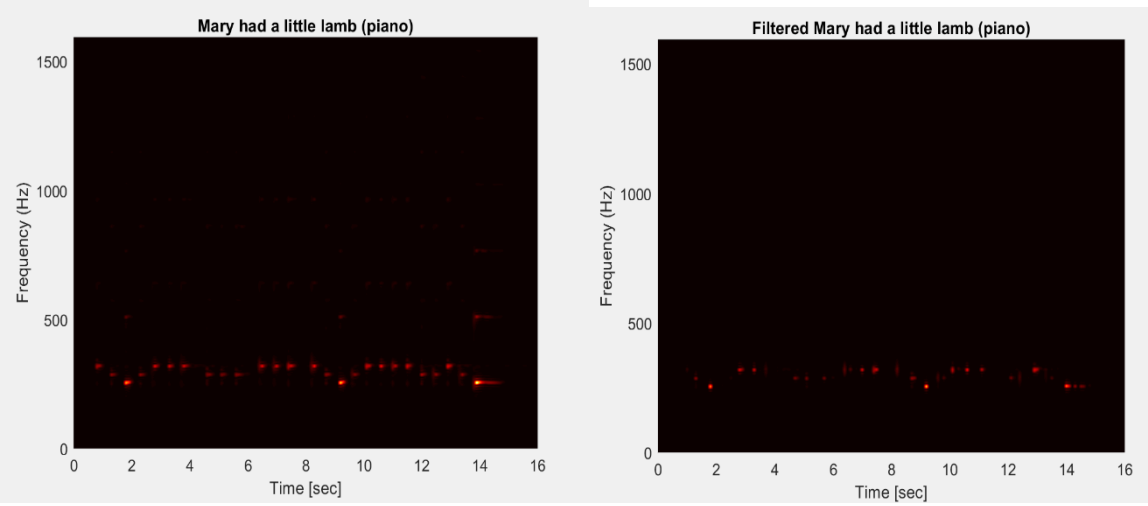


Figure 4. Piano Spectrogram of Unfiltered vs. Filtered Signal

Figure 5 shows the spectrogram generated from “music2.wav”, which is the same song played in the recorder. The overtones of this musical instrument are more faint than that of the piano, which is the reason why each instrument sounds different. The filtered signal can be found on the left graph of Figure 5. The complete music score is reproduced in the bottom graph of Figure 6.

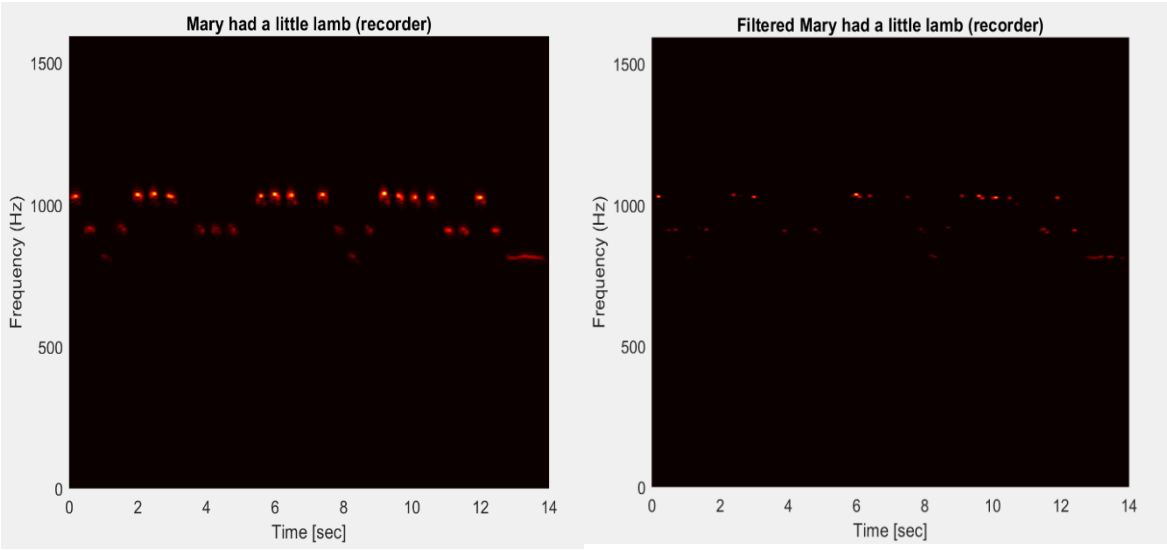


Figure 5. Recorder Spectrogram of Unfiltered vs. Filtered Signal

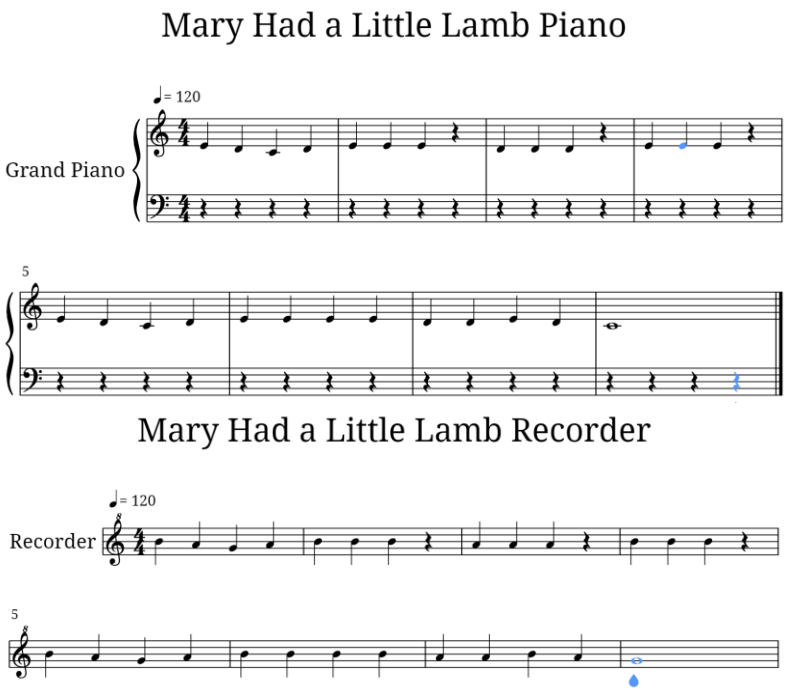


Figure 6. Musical Score of Mary Had a Little Lamb

Section 5. Summary and Conclusion

This assignment demonstrates the usage of Gabor Filtering to obtain spectrograms that conveys frequencies at each time slice. Increasing the time window of the wavelet results in increasing the frequency resolution of the spectrogram while sacrificing the time resolution. Lowering the sampling time (making the increments smaller) causes higher resolution in time, while sacrificing the clarity of frequencies which is caused by oversampling.

Analyzing the spectrogram of musical scores created using piano and recorder provided knowledge about the character of each instrument through their overtones. We observed that piano has stronger overtones that recorder through the spectrogram. Finally, by looking up the frequencies (in Hz) of the spots in the spectrogram, we can reproduce the musical sheet of the song Mary Had a Little Lamb provided.

One extra feature I added to the code is the filtering of the recording in the time domain, so that we can play the filtered version of the recording, causing unique sounding music scores.

This technique is extremely useful if we are interested in recovering characteristic frequencies in each time slice. Previously in homework 1, we only recover one frequency for the whole signal. In homework two, we recover various characteristic frequencies throughout the whole signal.

Appendix A: MATLAB Functions Implementation Explanation

audioplayer(v, Fs)

Creates an audioplayer object using amplitudes specified in v and the sampling rate specified in Fs.

playblocking(p8)

Plays the audioplayer object p8.

pcolor(tslide, fftshift(k), spc.'), shading interp, colormap(hot)

Creates a spectrogram using the X values specified in tslide, Y values specified in fftshift(k), and the color specified by the amplitude data specified in spc.'. The color corresponding to decimal values are determined by interpolation and the color are specified by the map: hot.

fft(vf)

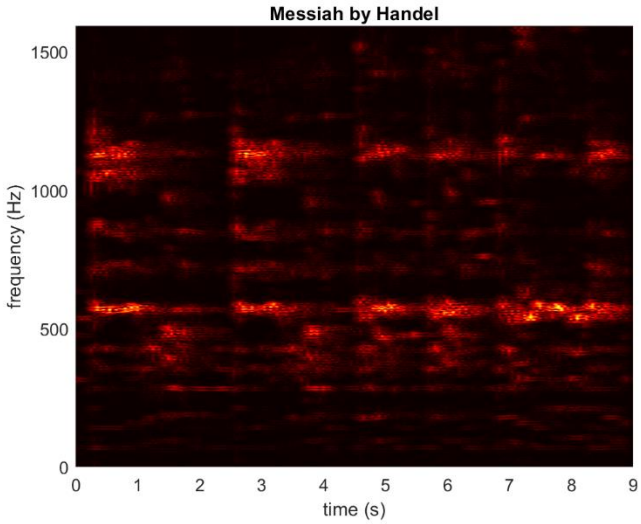
Computes the discrete Fourier transform with computational complexity of O(NlogN). The domain of the fourier transform is 2π.

fftshift(vft)

Rearranges the fourier transformed signal so that it is not flipped around the Y-axis.

ind = ind2sub(n, find(vft == max(vft)))

Returns the corresponding value of a data from the given index (the index of the maximum value in the vector in this case).



Bonus Graph: Mexican Hat Filter Spectrogram

Appendix B: MATLAB Code

```
% AMATH 482 - Homework 2
% Khrisna Kamarga

%% Part I
clear all; close all; clc;

% 1. Through use of the Gabor filtering we used in class, produce
%    spectrograms of the piece of work.
% 2. Explore the window width of the G´abor transform and how it
%    effects the spectrogram.
% 3. Explore the spectrogram and the idea of oversampling (i.e. using
%    very small translations of
%    the G´abor window) versus potential undersampling
%    (i.e. using very course/large translations of the G´abor window).
% 4. Use different G´abor windows. Perhaps you can start with the
%    Gaussian window, and look to see how the results are effected with
%    the Mexican hat wavelet and a step-function (Shannon)window.

load handel % load the "Halleluya" song
v = y'/2; % convert into row vector, attenuate the volume by half
v(end) = []; % since the length is not even, take out the last element
plot((1:length(v))/Fs,v); % plotting the song
t = (1:length(v))/Fs; % making time vector (length Fs)
L = max(t); n = length(t); % prepare the variables for fft
k=(2*pi/L)*[0:n/2-1 -n/2:-1];
xlabel('Time [sec]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');
p8 = audioplayer(v,Fs); % play the song
playblocking(p8); % ???

%% Gabor Filtering Animation
tslide = 0:0.05:9; % sampling time
twindow = 12; % the width of the super gaussian (it's yuuuge)
for j = 1:length(tslide) % each sample
    g = exp(-(twindow*(t-tslide(j))).^10); % super gaussian
%     sigma = 0.05;
%     g = 2/(sqrt(3*sigma)*pi^0.25)*(1-((t-tslide(j))/sigma).^2).*exp(-(t-
tslide(j)).^2/(2*sigma^2));
    vf = g.*v; % gabor filtering
    vft = fft(vf); % fourier transform of each wavelet
    subplot(3,1,1)
    plot(t,v,'k-',t,g,'r-') % the original song
    subplot(3,1,2)
    plot(t,vf,'k-') % each wavelet
    axis([0 9 -0.5 0.5])
    subplot(3,1,3)
    plot(fftshift(k),abs(fftshift(vft)/max(abs(vft))),'b-');
    % fourier wavelet
    pause(0.01)
end

%% Spectrogram
clc; close all;
t_sample = 0.05; %sampling rate
%0.05

tslide = 0:t_sample:9; % sampling time
twindow = 2; % the width of the super gaussian (it's yuuuge)
```

```
%10000000000000
%sigma = 0.05 sometimes 0.1
spc=[]; %matrix of all the wavelets
for j=1:length(tslide)
    g = exp(-(twindow*(t-tslide(j))).^10); % super gaussian
%     sigma = 0.05;
%     g = 2/(sqrt(3*sigma)*pi^0.25)*(1-((t-tslide(j))/sigma).^2).*exp(-(t-
tslide(j)).^2/(2*sigma^2));
    vf=g.*v;
    yft=fft(vf);
    spc=[spc;abs(fftshift(yft))];
end
% spectrogram
pcolor(tslide,fftshift(k)/(2*pi),spc.'), shading interp, colormap(hot)
title("Messiah by Handel");
xlabel("time (s)");
ylabel("frequency (Hz)");
axis([0 9 0 1e+04/(2*pi)])

%% Part II
clear all; close all; clc;

% 1. Through use of the G´abor filtering we used in class,
%    reproduce the music score for this simple piece. See Fig. 1 which has
%    the music scale in Hertz. (note: to get a good clean score, you
%    may want to filter out overtones... see below).
% 2. What is the difference between a recorder and piano? Can you see
%    the difference in the timefrequency analysis? Note that many people
%    talk about the difference of instruments being related to the
%    timbre of an instrument. The timbre is related to the overtones
%    generated by the instrument for a center frequency. Thus if you are
%    playing a note at frequency ?0, an instrument will generate overtones
%    at 2?0, 3?0, · · · and so forth.

% Piano

tr_piano=16; % record time in seconds
y=audioread('music1.wav'); Fs=length(y)/tr_piano;
v = y'/2;
plot((1:length(v))/Fs,v);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (piano)'); drawnow
p8 = audioplayer(y,Fs); playblocking(p8);
%% Spectogram
t = (1:length(v))/Fs; % time vector
L = max(t); n = length(t); % prepare for fft
k=(2*pi/L)*[0:n/2-1 -n/2:-1]; % wave numbers

tslide = 0:0.1:max(t); % sampling time
twindow = 10000000000000; %the width of the super gaussian
tau = 0.00001; % gaussian filter width

spc=[]; % all gabor wavelets
spcf=[]; % all filtered gabor wavelets
for j=1:length(tslide)
    g = exp(-twindow*(t-tslide(j)).^10); % super gaussian
    vf=g.*v; % gabor wavelets
    vft=fft(vf); % fourier transform of the wavelets
    ind = ind2sub(n, find(vft == max(vft))); % finding the strongest freq.
    kc = k(ind); % the corresponding wave number for the strongest freq.
    filter = exp(-tau*((k - kc).^2)); % gaussian filter
    vftf=filter.*vft; % filtered freq. domain wavelets
    spc=[spc;abs(fftshift(vft))];
```

```
    spcf=[spcf;abs(fftshift(vftf))];
end

figure(1)
pcolor(tslide,fftshift(k)/(2*pi),1000*abs(spc.')), shading interp, colormap(hot)
axis([0 max(t) 0 10000/(2*pi)]);
xlabel('Time [sec]'); ylabel('Frequency (Hz)');
title('Mary had a little lamb (piano)');
figure(2)
pcolor(tslide,fftshift(k)/(2*pi),1000*abs(spcf.')/max(max(abs(spcf.')))), shading interp,
colormap(hot)
axis([0 max(t) 0 10000/(2*pi)]);
xlabel('Time [sec]'); ylabel('Frequency (Hz)');
title('Filtered Mary had a little lamb (piano)');
%% Recovering the Filtered Time Domain Song
clc;
tau = 0.00001;
tspan = max(t)/length(tslide);
tpatch = 0:tspan:max(t);

v_filtered = zeros(size(k)); % filtered time domain song
for l = 1:length(tslide)
    g = exp(-twindow*(t-tslide(l)).^10);
    vf=g.*v;
    vft=fft(vf);
    ind = ind2sub(n, find(vft == max(vft)));
    kc = k(ind);
    filter = exp(-tau*((k - kc).^2));
    vftf=filter.*vft;
    vff = real(ifft(vftf)); %real part of the filtered time domain song
    v_filtered = v_filtered + vff;
end
%%
p8 = audioplayer(v_filtered*10,Fs); playblocking(p8);
%% Recorder
clear all; clc; close all;

tr_rec=14; % record time in seconds
y=audioread('music2.wav'); Fs=length(y)/tr_rec;
v = y'/2;
plot((1:length(v))/Fs,v);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (recorder)'); drawnow
p8 = audioplayer(y,Fs); playblocking(p8);
%% Spectogram
t = (1:length(v))/Fs; % time vector
L = max(t); n = length(t); % prepare for fft
k=(2*pi/L)*[0:n/2-1 -n/2:-1]; % wave numbers

tslide = 0:0.1:max(t); % sampling time
twindow = 10000000000000; %the width of the super gaussian
tau = 0.001; % gaussian filter width

spc=[]; % all gabor wavelets
spcf=[]; % all filtered gabor wavelets
for j=1:length(tslide)
    g = exp(-twindow*(t-tslide(j)).^10); % super gaussian
    vf=g.*v; % gabor wavelets
    vft=fft(vf); % fourier transform of the wavelets
    ind = ind2sub(n, find(vft == max(vft))); % finding the strongest freq.
    kc = k(ind); % the corresponding wave number for the strongest freq.
    filter = exp(-tau*((k - kc).^2)); % gaussian filter
    vftf=filter.*vft; % filtered freq. domain wavelets
```

```
    spc=[spc;abs(fftshift(vft))];
    spcf=[spcf;abs(fftshift(vftf))];
end
%%
figure(1)
pcolor(tslide,fftshift(k)/(2*pi),1000*abs(spc.')), shading interp, colormap(hot)
axis([0 max(t) 0 10000/(2*pi)]);
xlabel('Time [sec]'); ylabel('Frequency (Hz)');
title('Mary had a little lamb (recorder)');
figure(2)
pcolor(tslide,fftshift(k)/(2*pi),1000*abs(spcf.')/max(max(abs(spcf.')))), shading interp,
colormap(hot)
axis([0 max(t) 0 10000/(2*pi)]);
xlabel('Time [sec]'); ylabel('Frequency (Hz)');
title('Filtered Mary had a little lamb (recorder)');
%% Recovering the Filtered Time Domain Song
clc;
tau = 0.001;
tspan = max(t)/length(tslide);
tpatch = 0:tspan:max(t);

v_filtered = zeros(size(k)); % filtered time domain song
for l = 1:length(tslide)
    g = exp(-twindow*(t-tslide(l)).^10);
    vf=g.*v;
    vft=fft(vf);
    ind = ind2sub(n, find(vft == max(vft)));
    kc = k(ind);
    filter = exp(-tau*((k - kc).^2));
    vftf=filter.*vft;
    vff = real(ifft(vftf)); %real part of the filtered time domain song
    v_filtered = v_filtered + vff;
end
%%
p8 = audioplayer(v_filtered*10,Fs); playblocking(p8); %filtered song
```