AMATH482: Gabor Transform

Shuying Zhang
email: shuying3@uw.com

Abstract

This project explores the application of Gabor Transform through two classic music pieces: Hallelujah and Mary had a little lamb. Based on the spectrograms got by varying the width and the translation rate of the Gabor windows, how this two parameters will influence the spectrograms is clearly shown. The signature frequency is also identified through Gabor Transforms.

1 Introduction and Overview

We know that Fourier Transform does a great job in analyzing signals. On the other hand, it fails to return the time information of the given signals. Gabor Transform, also known as short-time Fourier Transform, overcomes this drawback of the Fourier Transform through sliding Gabor windows. This project will look into the applications of Gabor Transformation using three classic music work and see how window parameters will influence the result of the Gabor Transforms.

In the problem given, the music pieces contain a 9 seconds Hallelujah and two Mary Had A Little Lamb with length 16 seconds and 14 seconds in piano and recorder respectively. We applied Gabor Transforms on each piece.

There are five sections in this report. The introduction and overview section gives a brief description of the problem and the report. The theoretical part provides background knowledge for Gabor Transform and the window we will utilize. In algorithm implementation and development, we will formulate the problem into two parts: exploring the Hallelujah spectrograms and identifying the Mary Had A Little Lamp key scores in MATLAB. The computational results shows the plots of music pieces after Gabor Transforms. A summary is concluded in summary and conclusions. All the MATLAB code and related MATLAB commands are in appendix.

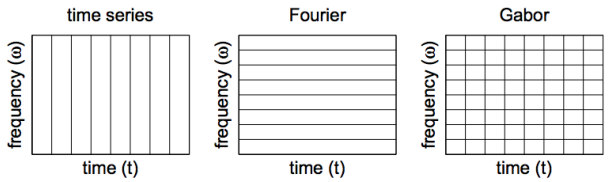

Figure 1: Difference among Time-Frequency Series Analysis; Fourier Analysis and Gabor Analysis.

2 Theoretical Background

We will go over what is Gabor Transform and how will it help us in showing signature frequency in time. Some descriptions and figures are from the class notes [1].

2.1 Gabor Transform

Gabor Transform is also called short-time Fourier Transform because it uses the idea of a sliding window and in that window, perform Fourier Transform to identify the center frequency. In this way, Gabor Transform gives us the frequencies without lose any moment in time. The signals we are analyzing are not limited to stationary ones only since now we could localize time and frequency in time-frequency analysis.

Gabor introduces the new kernel

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

Gabor Transform is defined as:

$$\mathcal{G}[f](t,\omega) = \tilde{f}_g(t,\omega) = \int_{-\infty}^{\infty} f(\tau)\overline{g}(\tau - t)e^{-i\omega\tau} d\tau = \left(f, \overline{g}_{t,\omega}\right) \tag{2}$$

We can see that Gabor Transform extracts and time and frequency content from a signal. Figure 1 summarizes the difference among three signal analysis.

2.2 Gabor Windows

In Gabor Transform, the term $g(\tau - t)$ is a time filter that localizes the signal over a specific window of time where τ is the parameter of the slides. Figure 2 is an illustration of sliding using Gaussian windows. $g(\tau - t)$ means the Fourier Transform is centered at t = τ.

For convenience, Gabor filters are considered to be: $\|g(\tau - t)\| = 1$ and $\|g(t)\| = 1$.

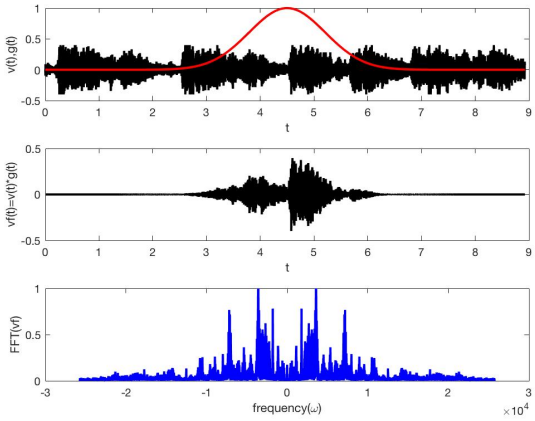

Figure 2: Window Sliding: First row: red curve shows what the window (filter) looks like at time = t; Second row: the signal after filtering at time = t; Third row: The filtered signal in Fourier Space.

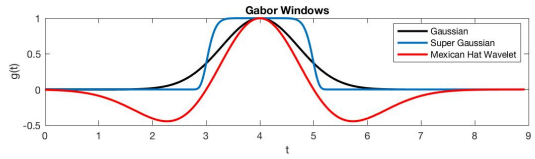

Figure 3: Three commonly used Gabor Windows

One common filter used here can be Gaussian filter:

$$g(t) = -e^{a*(t-\tau).^2} \tag{3}$$

where a and τ are introduced above.

We can change power exponent from 2 to 10 to make it into a super Gaussian filter. See the steel blue curve in Figure 3 to see a super Gaussian filter centered at 4.

Wavelets To improve Gabor Transform, there is a theory of wavelets in windowed Fourier Transforms including mother and father wavelets. Here the width a and the time τ are defined to be the scaling and the translation of a signal.

For example, mother wavelet is defined as:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \tag{4}$$

One wavelet we used as a filter in the real code is called Mexican Hat Wavelet. See the red curve in Figure 3 to see Mexican Hat $\psi_{1,0}$. It's defined as:

$$\psi(t) = \left(1-t^2\right)e^{-t^2/2} \tag{5}$$

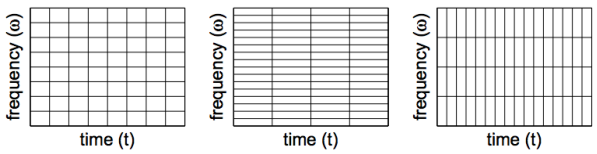

Figure 4: Trade-offs Between Frequency and Time: Equal localization (Left); Longer window, better frequency, poor time resolution(middle); Shorter window, better time resolution, poor frequency (Right)

This is actually the second moment of Gaussian filter.

2.3 Drawbacks of Gabor Transform

Although Gabor Transform discretizes the time and frequency domain and successfully localizes time. However, it has some limitations by time filtering itself. In Figure 4, we are able to see the trade-offs in Gabor Transform. The time window allows us to filter the signals at time τ with width a. This fails to capture any frequency with wave length longer than the width. This indicates that if we increase the width of the window, frequency resolution is better but time resolution will decrease; but if we shorten the window width, time resolution increases, whereas the frequency resolution becomes poor.

3 Algorithm Implementation and Development

3.1 Starting the problem

Preparation We plot the piece of Hallelujah and Mary Has A Little Lamb to visualize the what the signals look like.

In this project, we set the Fourier mode n to be the size of the signal and domain to be the length of the signal (in second). Previously, we said the number of nodes must be 2^n. Here `fft` command will accommodate that. First use *linspace* to divide domain [0, L]. Then, rescale the domain by multiplying $\frac{2\pi}{L}$. What's more, we also need to shift k to center using *fftshift*. This shift is necessary for plotting all data in Fourier space.

3.2 Hallelujah

Spectrogram through Gabor Filtering To use Gabor transform, we need to pick a window and set the width and the translation rate. Gaussian filter is a commonly utilized window. Initially, we set the width to be 1 and translation rate to be 0.1 to explore the signal.Thus tslide = 0: 0.1: L.

Window Width and How It Affects Spectrograms Here, we will explore the relationship between the frequency resolution and the time resolution, namely , the width of the window. The test width are defined to be 1, 10, 0.2, and 100. Among them, 0.2 and 100 are picked deliberately to show the relationship clearly.

Oversampling (small translation of window) and Undersampling (large translation window) Here, we will explore the relationship between the frequency resolution and the rate of the translation. This means we need to vary the sliding speed for now.

The test translation rates are 0.05, 0.1, 0.5, and 1. We can't choose any smaller rate due to the run time and computational limitations, and a step of 1 is large enough to refelct the drawbacks.

Different windows We apply different windows other than Gaussian in this section, including Super Gaussian and Mexican Hat Wavelets.

3.3 Mary Had A Little Lamb

To explore the difference between a piano and a recorder and navigate the simple score frequency. We will plot the spectrograms like what we do in the section above. L and signal are loaded similar to the process in preparation. Here, to better display the result, we will set the window width to be 50. A smaller window doesn't give a desired frequency resolution.

Besides, there is one more step when using pcolor command. Since $\omega = 2\pi f$, in order to plot the note in Hertz, we need to rescale y-axis in Hertz by dividing 2π.

To focus on the clear scores only, we zoom in to filter out any overtones.

4 Computational Results

At first, the signal of Hallelujah is shown in Figure 5.

After applying a Gaussian filter with width = 1, and translation rate = 0.1, the spectrogram gives us the result in Figure 6.

In order to find how window width will affect spectrograms, by fixing the translation rate = 0.1. We get the following result in Figure 7.

The result fits what we discussed in the Theoretical Background Part: For shorter windows (See Width = 0.02), we got good time resolution but poor frequency resolution; For larger windows (See Width = 100), Frequency resolution are better by sacrificing time resolution.

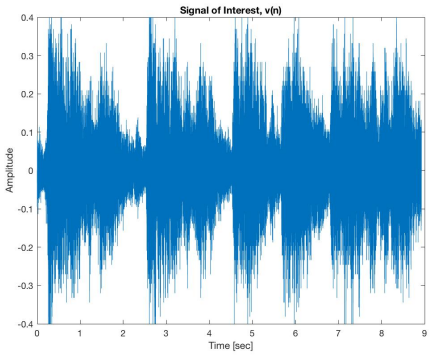

Figure 5: Hallelujah Signals

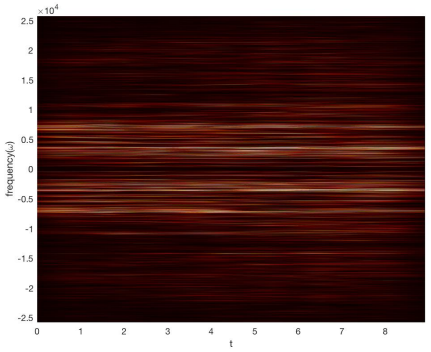

Figure 6: Spectrograms With Width = 1 and Translation = 0.1

To find out the idea of over sampling, we fix the window width to be 20 and change the tranlation rate instead. The result is in Figure 8.

Per my observation, when the translation rate is small (See Translation = 0.05), the frequency resolution behaves normally. No big difference spotted by comparing it with Translation = 0.1 figure. However, the run time grows dramatically as we have more windows to process. The excessive number of windows doesn't gives us a better result but spends more time. This is known as oversampling. When the translation rate increases in big steps like 1 (See Translation = 1), discontinuity appears in the result. Hence, the result is not complete. This is known as undersampling.

At last, for Hallelujah, different windows returns the result in Figure 9. We set the width and translation rate uniformly as 20 and 0.1 respectively to display the patterns in Figure 9.

The result shows that compared with Gaussian window, super Gaussian does a better job in time resolution but poor in frequency, and Mexican Hat Wavelets has a better time and frequency resolution overall.

The inital signal of two versions of Mary Had A Little Lamb are in Figure 10

For the piano version of Mary Had A Little Lamb after Gabor Transform with Gaussian

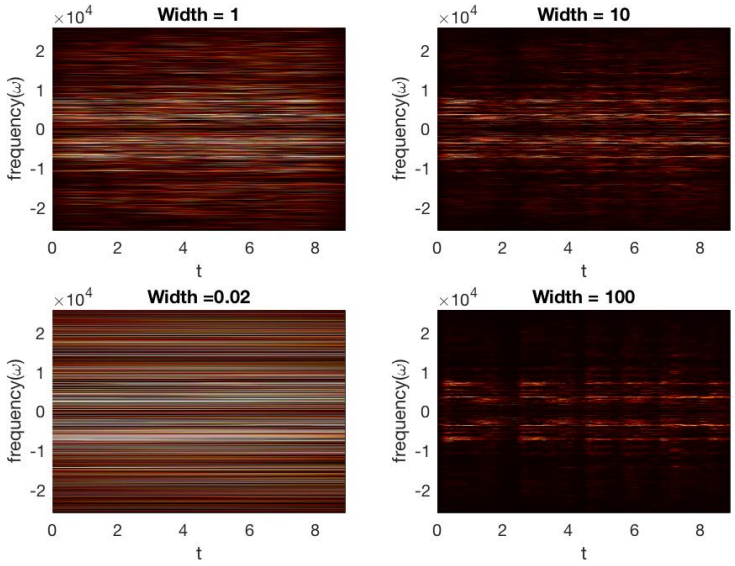

Figure 7: Spectrograms by Varying Window Width

filter of width = 50 and translation rate = 0.1, all the frequencies extracted are in Figure 11.

Above the clear scores, we can see some other scores in higher Hertz, like $2f$ and $3f$. These overtones are generated due to the timbre of the piano. To filter out those overtones, we zoom in y-axis to see the clear scores in Figure 12.

For the recorder version of Mary Had A Little Lamb, after Gabor Transform with Gaussian filter of width = 50 and translation rate = 0.1, all the frequencies extracted are in Figure 13.

In general, the recorder produces higher frequencies than the piano. There are no overtones but some frequencies lower than the signature frequency present right below the clear scores.

5 Summary and Conclusions

In this project, we apply Gabor Transform on two music pieces. Through this process, we explore the frequency in time spectrograms using different Gabor windows and discuss the factors that affect the spectrograms. At last, we successfully reproduce the scores in Mary Had A Little Lamb pieces using different instruments through Gabor filtering.

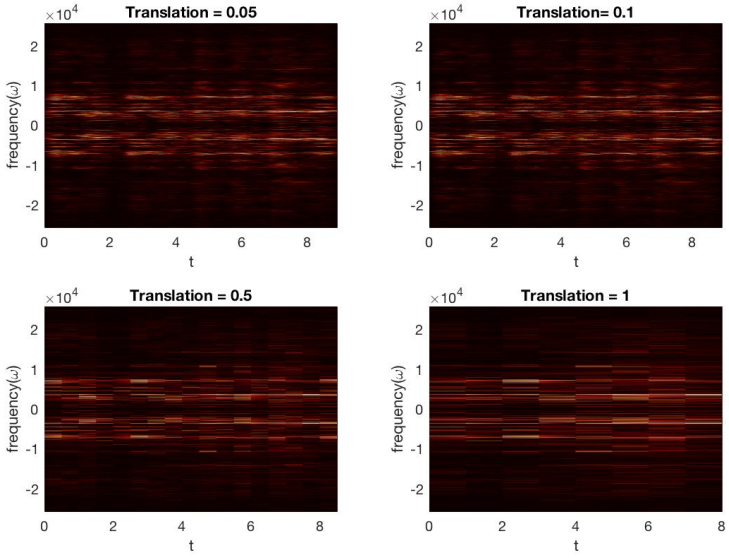

Figure 8: Spectrograms by Varing Window Translation Rate

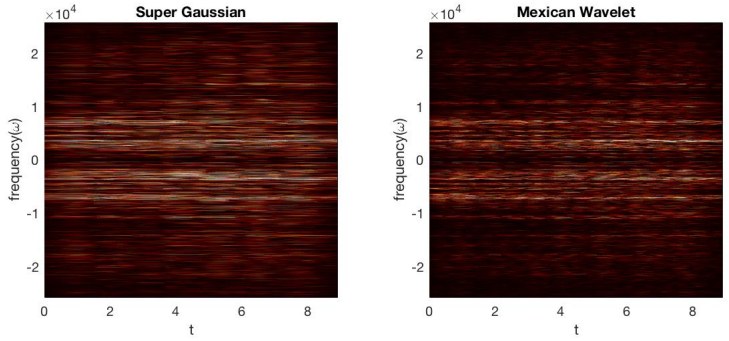

Figure 9: Different Gabor Windows on Hallelujah

References

[1] Part3 : Computation Methods for Data Analysis. Data-Driven Modeling & Scientific Computation: Methods for Complex Systems Big Data, by J. Nathan Kutz, OUP Oxford, 2013, available at `http://faculty.washington.edu/kutz/582.pdf`.

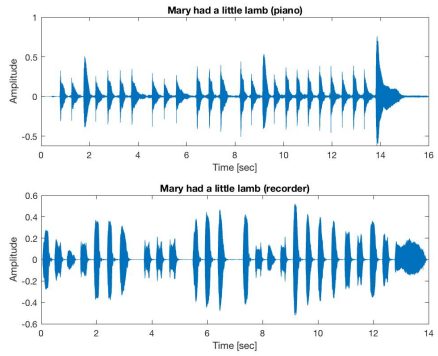

Figure 10: Mary Had A Little Lamb Signals

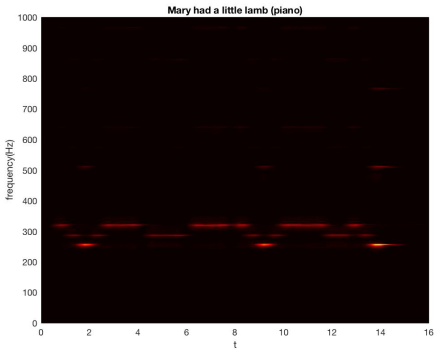

Figure 11: Mary Had A Little Lamb (piano) scores with overtones

A MATLAB commands

Documentation for those commands can be found in MATLAB using *help command_name*.

A.1 Data Modification

linspace(-L,L,n+1) linearly divides [-L,L] into n equally-spaced parts, returns values of n points in between.

A.2 FFT related

fftn(V) uses FFT to do fourier transform on n-dim data V. n can be replaced into any dimensions, like fft, fft2.

fftshift(V) shifts the FFT data V to center.

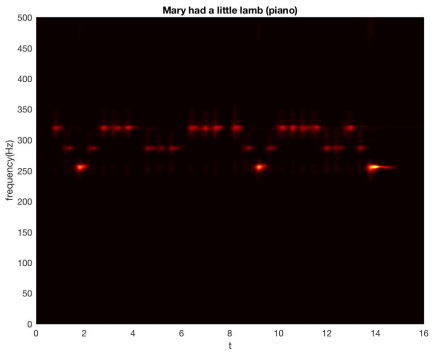

Figure 12: Mary Had A Little Lamb (piano)

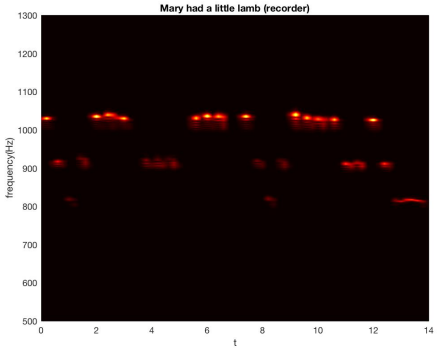

Figure 13: Mary Had A Little Lamb (recorder)

A.3 Plotting

pcolor(X,Y,C) X and Y are matrices where C is the value in each cell.

colormap(hot) color sets.

plot(X,Y) plots 2-dim data.

subplot(i,j,k)) Draw multiple plots in one figure. i and j specifies the lay out, k is the index of the plots.

B MATLAB code

```
clear all; close all; clc;
%% Part I
%Analyzing Handel's Messiah
```

```
%plot the frequency vs time
load handel
v = y'/2;
plot((1:length(v))/Fs,v);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');
%play the music piece
% p8 = audioplayer(v,Fs);
% playblocking(p8);

%initial setup
%fft
v = v(1:length(v) - 1);
L = length(v)/Fs;
n = length(v);
t2 = linspace(0,L,n+1); t = t2(1:n);
k = (2*pi/L)*[0:n/2-1 -n/2:-1]; ks = fftshift(k);
tslide = 0:0.1:L;

 % window width
spc = [];
%spectrogram through Gabor filtering
for i = 1:length(tslide)
    %filter
    g = exp(-(t-tslide(i)).^2); %Gaussian
    vf = g .* v ; % v filtered
    vft = fft(vf); % v filtered transformed

    %show sliding
%     subplot(3,1,1)
%     plot(t,v,'k-',t,g,'r-','linewidth',[2])
%     xlabel('t')
%     ylabel('v(t),g(t)')
%     subplot(3,1,2)
%     plot(t,vf,'k-','linewidth',[2])
%     xlabel('t')
%     ylabel('vf(t)=v(t)*g(t)')
%     subplot(3,1,3)
%     plot(ks,abs(fftshift(vft))/max(abs(vft)),'b-','linewidth',[2])
%     xlabel('frequency(\omega)')
%     ylabel('FFT(vf)')
%     pause(0.2)

    %store frequency
     spc = [spc; abs(fftshift(vft))];
```

```
end
pcolor(tslide, ks, spc.'), shading interp, colormap(hot) %plotting
xlabel('t')
ylabel('frequency(\omega)')

% % window width and how it affects spectrogram
width = [1;10;0.02;100]; %change window width
for k = 1:length(width)
    spc = [];
    for j = 1:length(tslide)
        g = exp(-width(k)*(t-tslide(j)).^2); %Gaussian
        vf = g .* v ;
        vft = fft(vf);
        spc = [spc; abs(fftshift(vft))];
    end
    subplot(2,2,k)
    pcolor(tslide, ks, spc.'), shading interp, colormap(hot) %plotting
    xlabel('t')
    ylabel('frequency(\omega)')
end

%oversampling (small translation of window) and under sampling (large
    translation window)
inc = [ 0.05; 0.1; 0.5; 1];
for p = 1:length(inc) % change window translation
   tslide = 0 : inc(p) : L;
   spc = [];
   for j = 1:length(tslide)
       g = exp(-20*(t-tslide(j)).^2); %Gaussian
       vf = g .* v ;
       vft = fft(vf);
       spc = [spc; abs(fftshift(vft))];
   end
    subplot(2,2,p)
    pcolor(tslide, ks, spc.'), shading interp, colormap(hot) %plotting
    xlabel('t')
    ylabel('frequency(\omega)')
end

% different windows
%Gaussian
gn = exp(-(t-4).^2);
%super Gaussian
gs = exp(-(t-4).^10);
%Mexican wavelet
mw = (1-(t-4).^2).*exp((-(t-4).^2)/2);
%plot(t,gn,'k-',t,gs,t,mw,'r-','Linewidth',[2])
```

```
spcgs = [];
spcmw = [];
tslide = 0:0.1:L;
for i = 1:length(tslide)
    %filters
    gs = exp(-(t-tslide(i)).^10);
    mw = (1-10*(t-tslide(i)).^2).*exp((-10*(t-tslide(i)).^2)/2);
    %filter: Super Gaussian
    vfgs = gs .* v ; % v filtered
    vftgs = fft(vfgs); % v filtered transformed

    %filter: Mexican Wavelet
    vfmw = mw .* v ; % v filtered
    vftmw = fft(vfmw); % v filtered transformed
    %store frequency
     spcgs = [spcgs; abs(fftshift(vftgs))];
     spcmw = [spcmw; abs(fftshift(vftmw))];
end
subplot(1,2,1)
pcolor(tslide, ks, spcgs.'), shading interp, colormap(hot) %plotting
xlabel('t')
ylabel('frequency(\omega)')
title('Super Gaussian')
subplot(1,2,2)
pcolor(tslide, ks, spcmw.'), shading interp, colormap(hot) %plotting
xlabel('t')
ylabel('frequency(\omega)')
title('Mexican Wavelet')

% %% Part 2
% clear all; close all; clc;
tr_piano=16; % record time in seconds
y=audioread('music1.wav'); Fs=length(y)/tr_piano;
L = 16;
y = y';
n = length(y);
t2 = linspace(0,L,n+1); t = t2(1:n);
k = (2*pi/L)*[0:n/2-1 -n/2:-1]; ks = fftshift(k);
tslide = 0:0.2:L;
spc1 = [];
%filter:
for i = 1:length(tslide)
    g = exp(-50*(t-tslide(i)).^2);
    % filtered
    yg = g .* y;
    ygt = fft(yg);
```

```
    spc1 = [spc1; abs(fftshift(ygt))];
end
pcolor(tslide, ks/(2*pi), spc1.'), shading interp, colormap(hot) %plotting
axis ([0 16 0 500])
xlabel('t')
ylabel('frequency(Hz)')
title('Mary had a little lamb (piano)')
% subplot(2,1,1)
% plot((1:length(y))/Fs,y);
% xlabel('Time [sec]'); ylabel('Amplitude');
% title('Mary had a little lamb (piano)'); drawnow
% p8 = audioplayer(y,Fs); playblocking(p8);
%
tr_rec=14; % record time in seconds
y=audioread('music2.wav'); Fs=length(y)/tr_rec;
L = 14;
y = y';
n = length(y);
t2 = linspace(0,L,n+1); t = t2(1:n);
k = (2*pi/L)*[0:n/2-1 -n/2:-1]; ks = fftshift(k);
tslide = 0:0.2:L;
spc2 = [];
%filter:
for i = 1:length(tslide)
    g = exp(-50*(t-tslide(i)).^2);
    % filtered
    yg = g .* y;
    ygt = fft(yg);

    spc2 = [spc2; abs(fftshift(ygt))];
end
pcolor(tslide, ks/(2*pi), spc2.'), shading interp, colormap(hot) %plotting
axis ([0 14 500 1300])
xlabel('t')
ylabel('frequency(Hz)')
title('Mary had a little lamb (recorder)')
% subplot(2,1,2)
% plot((1:length(y))/Fs,y);
% xlabel('Time [sec]'); ylabel('Amplitude');
% title('Mary had a little lamb (recorder)');
% p8 = audioplayer(y,Fs); playblocking(p8);
```