Homework 2: Gabor Transform

Minghu Zhou

February 4, 2020

Abstract

There are two objectives in this project. First, we will find how the spectrums of Handel's *Hallelujah* is affected by different Gabor Transformations. Gabor Transformation can increase the resolution in time but decrease resolution in frequency and vice versa. It depends on what values we put in related parameters. Different filters can also affect both time and frequency resolution. Second, using Gabor transformation, we are able to decide the frequency difference between a melody of *Mary Had a Little Lamb* played by a piano and a recorder. We will find that the recorder actually plays a higher frequency than the piano and they don't have the same number of overtones even though the notes played are the same.

1. Introduction and Overview

We have many sounds of different frequencies in our life. Using Gabor transform we can filter out the noises in the sounds and analyze these sounds with graphs and numbers.

For the first part, we will use different parameters when applying Gabor Transformation to the song every time. These parameters include translation and width of the Gabor window and different types of filter. We can then render many spectrums and analyze the difference between them with respect to the time and frequency resolution.

In the second part, we try to use the Gabor Transformation to find the music score of a piece of the song. We will also observe the spectrums of the melody played by a recorder and a piano and determine their difference.

2. Theoretical Background

2.1 Gabor Transformation

Fourier Transformation is able to transform a signal in time domain to frequency domain. In particular, it can return all the frequencies in the whole time period, but it does not tell when the frequencies happen. Fortunately, the Gabor Transformation can tell us about when these frequencies occur.

Gabor Transformation of a function f is the following

$$G[f] = \tilde{f}_g(t,\omega) = \int_{-\infty}^{\infty} f(\tau)\mathrm{g}(\tau - \mathrm{t})\mathrm{e}^{-\mathrm{i}\omega\mathrm{t}} d\tau \qquad (2.1)$$

g is commonly a gaussian function with the form:

$$g_{t,\omega}(\tau) = \mathrm{e}^{\mathrm{i}\omega\mathrm{t}} g(\tau - t) \quad (2.2)$$

t corresponds to the center time.

The Gabor Transformation applies the Fourier function to a given function as it integrates over it. Practically, it moves a window which initially includes the beginning part of a signal and applies Fourier Transformation to that part. Then it moves the window forward, also shifting the filter forward. Now the window has another time segment to which can be applied Fourier Transformation. It repeats the process to the end of time. In this way, the change of frequencies over time is known.

One thing we need to know is that Gabor transform makes tradeoff between the time resolution and frequency. If we use a wider Gabor window, we will get more resolution on frequency but less on time. Contrarily, if we use a narrower window, we get more resolution on time but less on frequency. Further explanation will be given in the following sections.

2.2 Gabor Window Types

To remove the noise of a signal, we can use a filter to attenuate the signal not near the center time. There are many types of filter, we explore Gaussian window, Mexican hat wavelet and Shannon window.

Gaussian:

$$F(\tau) = e^{-a(\tau - t)^2} \quad (2.3)$$

Mexican hat:

$$F(\tau) = (1 - a(\tau - t)^2) * e^{-a(\tau - t)^2} \quad (2.4)$$

Shannon:

$$F(\tau) = \begin{cases} 1, & |a(\tau - t)| \leq 0.5 \\ 0, & |a(\tau - t)| > 0.5 \end{cases} \quad (2.5)$$

The above functions are centered at t and has a value of 1 at t. The bigger the a is, the narrower the window width is.

3. Algorithms Implementation and Development

3.1 Part one

We first load the data which includes a vector y and a variable Fs which corresponds to how many measurements are made per second for the song. Then, we divide the total

measurements which is length of y by Fs to get the length of the song. Now that we have the length of song L and the total measurement n, we can now construct the Fourier domain. Since n is odd, our modes will be in $[-\frac{n-1}{2}, \frac{n-1}{2}]$. We rescale frequencies by multiplying modes in $[-\frac{n-1}{2}, \frac{n-1}{2}]$ by 2π / L since the FFT assumes 2π periodic signals. We also have to swap the first half of the frequency domain and the second half as FFT function shifts the input data in time domain that way.

Let's start with using a normal Gabor filter to produce a spectrum. We will first use a Gaussian filter and choose a common width 1 for the filter. Then, we create a vector representing time slide which starts from 0 to the end of the time. Each time step in this vector will be a center for the filter in each iteration later. Through iterations over the time steps, we apply the filter to the signal and then use Fast Fourier Transform to transfer the signal to its frequency domain. We need to *fftshift* the absolute value of the result and store it in a matrix. After the iterations, we can then plot the spectrum and view how different frequencies change over time.

Next, we change some parameters of the Gabor Transformation to explore the resulting difference in spectrums. We will explore three parameters. First, we vary the window width; we will set a to be 0.1, 1, 10 in the Gaussian Filter we use where 0.1 corresponds to a large window width, 10 to a small and 1 to a normal. Second, we change the translation of the Gabor windows which represents the time step in the time slide vector. We set the time step to be 0.01, 0.1 and 1 where 0.01 may corresponds to oversampling and 1 to undersampling and 0.1 to normal sampling. Last, we will use different window types. In particular, we choose Gaussian filter, Shannon filter and Mexican hat wavelet and meanwhile keep the time step and the window width the same.

3.2 Part two

The process of the second part is almost the same to part one. We read two signals from two files, one is played by a piano and another by a recorder. Then we apply same Gabor Transformation to both of them and see the difference of the resulting spectrums. Since we also want to know the differences between their timbres, we keep the overtones in the spectrums to learn how the behavior of the overtones differ between the two instruments. Note that this time n is even, so our modes will be in $[-\frac{n}{2}, \frac{n}{2} - 1]$. This time, in order to find the music notes, we only need to multiply the modes by 1 / L since the frequency(f) of each note is recorded in Hz. The relationship between ω and f is $\omega = 2\pi f$, which means we need to divide the frequency by 2π to get frequency in Hz.

4. Computational Results

4.1 Part one

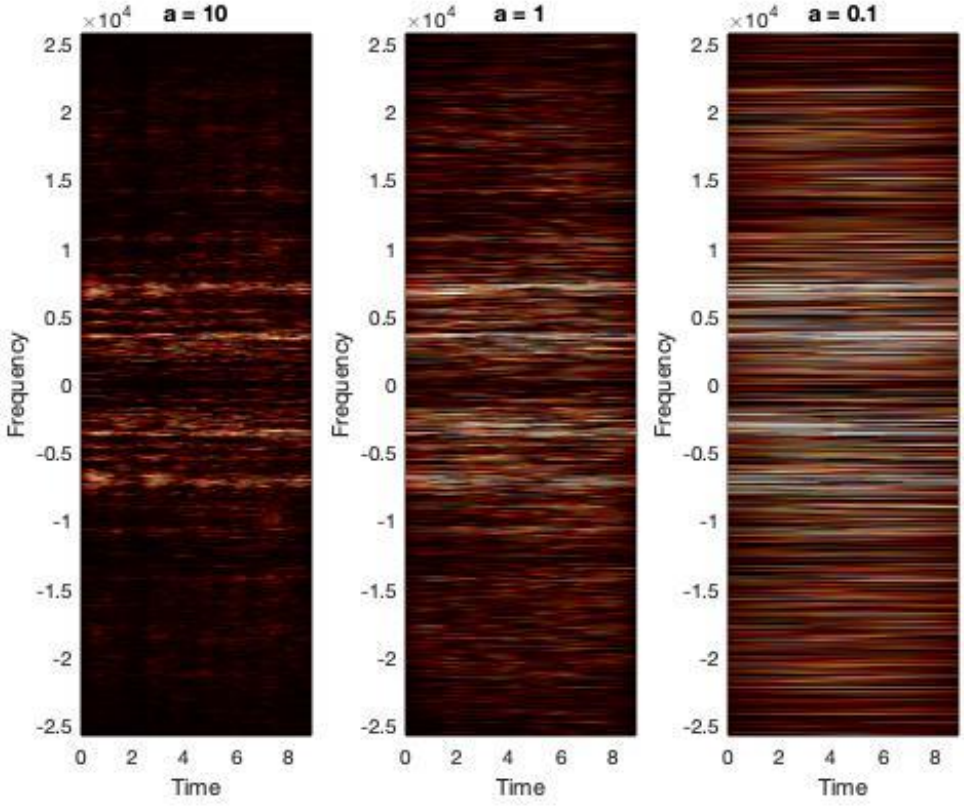

Figure 1: Spectrograms for varying Gaussian window width

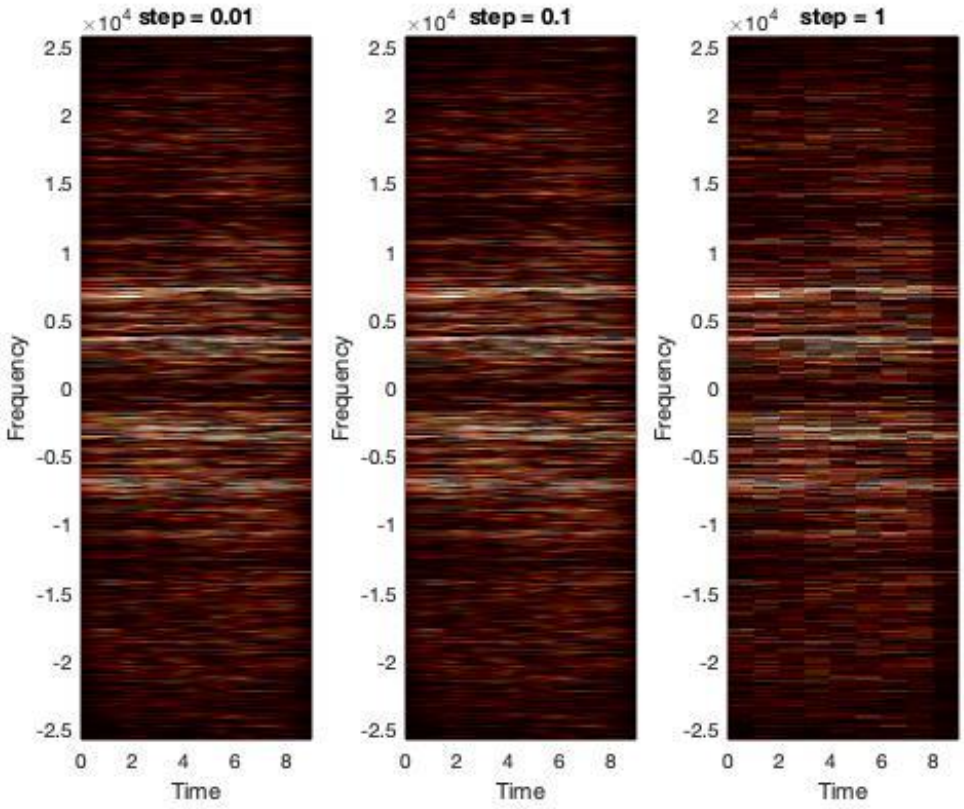

Figure 2: Spectrograms for varying Gaussian window translation

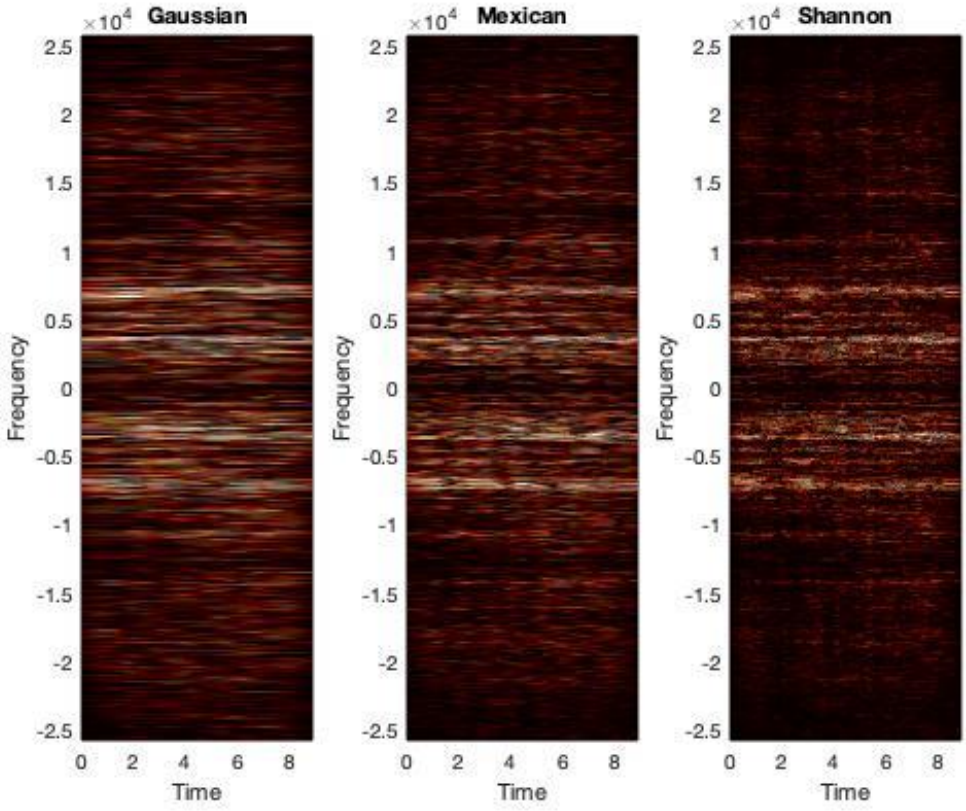

Figure 3: Spectrograms for different Gabor windows

From figure 1 we can see that a larger Gabor window width can increase the resolution in frequency but decrease resolution in time; we can clearly know what frequencies are included in the song but barely know when they are happening. A narrower window width allows us to know what frequencies are happening at a time but loses the information of some weaker frequencies.

From figure 2 we can see the smoothness of the spectrums changes as the Gaussian window translation changes. The spectrum with a larger time slide is more uneven compared to those with small time slide, which is a case of undersampling. However, we don't see much difference between their shapes and resolution of time and frequencies. When we make the time slide smaller and smaller, the resulting spectrum doesn't change very much but takes much time to generate so that is a kind of oversampling.

In figure 3, when using different filter types, we see that Gaussian filter has a best resolution in frequency worst while the Shannon filter has best resolution in time but worst resolution in frequency. We also see that spectrum for Shannon window has some sharp edges due to the square shape of the Shannon filter. Spectrums for Gaussian and Mexican hat looks better because of their smooth shape.

4.2 Part two

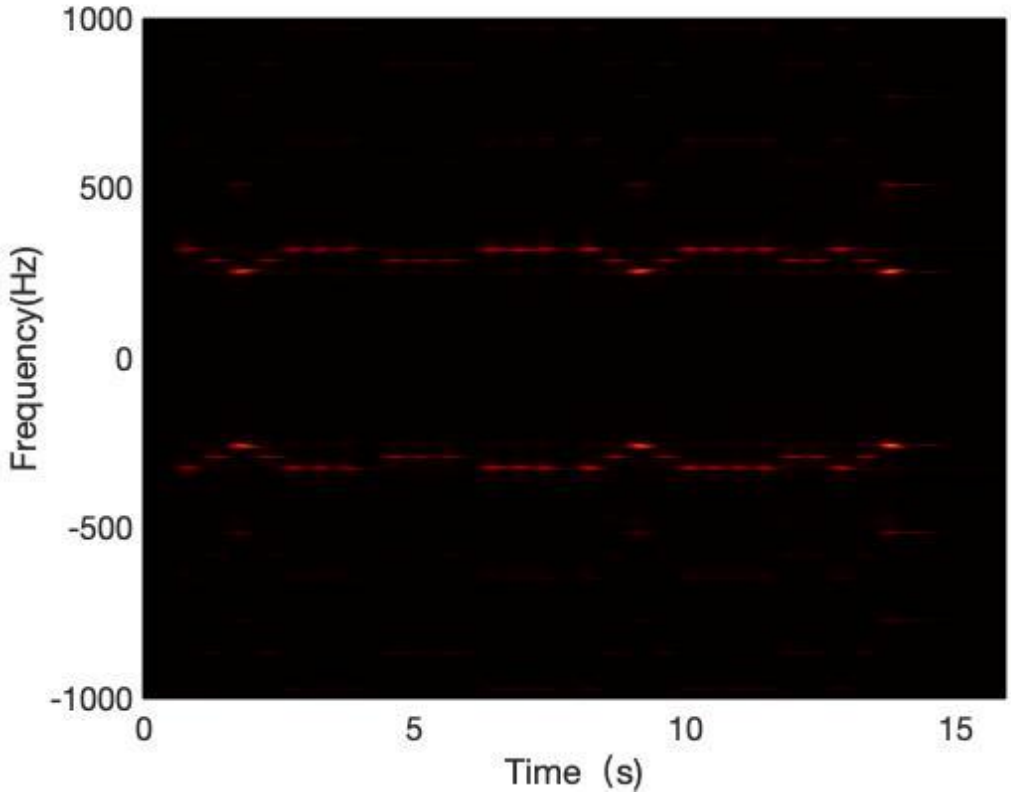

Figure 4: Spectrum of *Mary had a Little Lamb* played by a piano

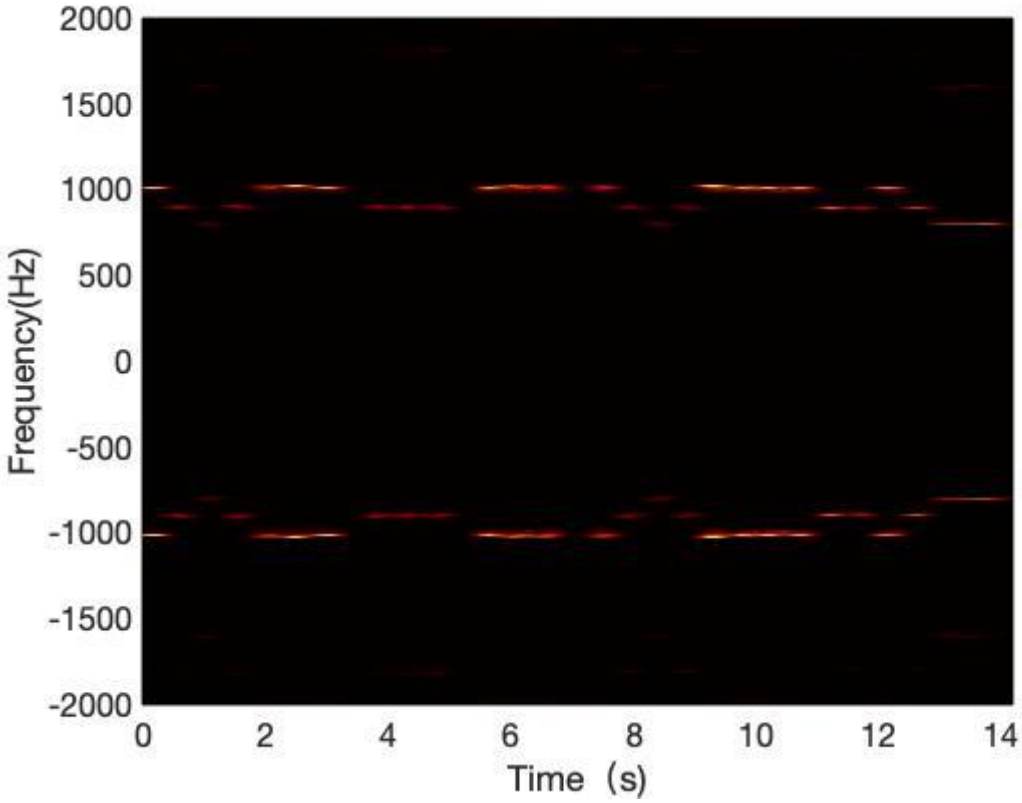

Figure 5: Spectrum of *Mary had a Little Lamb* played by a recorder

From The above figures we can see that the piano plays notes at lower frequencies than the recorder, which means the recorder plays the notes at higher pitches. We also see that there are more overtones generated by the piano than the recorder. For example, the last note in the music played by the piano has a frequency around 250Hz, and we can also observe some faint overtones around 500Hz and 750Hz. However, there are no such overtones in the spectrum of the music played by the recorder. These overtones can affect the timbre of the instrument. The above two points explain why we hear different sounds of the same piece of music.

5. Summary and Conclusions

We want to see how the frequencies change with respect to time of the input sound signal, so we use apply Gabor transform to make spectrums and visualize the data. We also explore how change of parameters in Gabor transformation affect the resulting spectrums. Through use of Gabor filtering, we reproduce the music score for a piece of music played by a recorder and a piano and explore how they are different by looking at the difference between their spectrums.

Appendix A.
MATLAB functions and brief implementation explanation

- $linspace(x1, x2, n)$: Return n points. The spacing between the points is (x2-x1)/(n-1).
- $fft(X)$: Computes the discrete Fourier transform (DFT) of X using a fast Fourier transform (FFT) algorithm.
- $fftshift(X)$: Rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array.

Appendix B.
MATLAB codes

Part 1:

```
clear all; close all;clc

%% set up
load handel
v = y';
% plot((1:length(v))/Fs,v);
% xlabel('Time [sec]');
% ylabel('Amplitude');
% title('Signal of Interest, v(n)');

n = length(v);
t = (1:n)/Fs;
v = v(1:n);
```

```
L = t(end);
k = 2*pi/L*[0:(n-1)/2 -(n-1)/2:-1];
ks = fftshift(k);
%% Spectrograms for varying Gaussian window width
tslide = 0:0.1:L;
a_vec = [10 1 0.1];

for j = 1:length(a_vec)
    a = a_vec(j);
    vgt_spec = zeros(length(tslide),n);
    for i = 1:length(tslide)
        g = exp(-a*(t-tslide(i)).^2);
        vg = g.*v;
        vgt = fft(vg);
        vgt_spec(i,:) = fftshift(abs(vgt));
    end
    subplot(1,3,j)
    pcolor(tslide,ks,vgt_spec.'),
    shading interp
    title(['a = ',num2str(a)])
    xlabel('Time'), ylabel('Frequency')

    colormap(hot)
end
%% Spectrograms for varying Gaussian window translation

steps = [0.01 0.1 1];
for j = 1:length(steps)
    step = steps(j);
    tslide = 0:step:9;
    vgt_spec = zeros(length(tslide),n);
    for i = 1:length(tslide)
        g = exp(-(t-tslide(i)).^2);
        vg = g.*v;
        vgt = fft(vg);
        vgt_spec(i,:) = fftshift(abs(vgt));
    end
    subplot(1,3,j)
    pcolor(tslide,ks,vgt_spec.'),
    shading interp
    title(['step = ',num2str(step)])
    xlabel('Time'), ylabel('Frequency')

    colormap(hot)
end
%% Spectrograms for different Gabor windows
tslide = 0:0.1:L;
name = ["Gaussian","Mexican","Shannon"];
for j = 1:3
    vgt_spec = zeros(length(tslide),n);
    for i = 1:length(tslide)
        if (j == 1)
            g = exp(-(t-tslide(i)).^2);
        elseif (j == 2)
            g = (1-(t-tslide(i)).^2).*exp(-(t-tslide(i)).^2);
        else
            g = abs(t-tslide(i)) <= 0.5;
        end
        vg = g.*v;
        vgt = fft(vg);
```

```
        vgt_spec(i,:) = fftshift(abs(vgt));
    end
    subplot(1,3,j)
    pcolor(tslide,ks,vgt_spec.'),
    shading interp
    title(name(j))
    xlabel('Time'), ylabel('Frequency')

    colormap(hot)
end
```

Part 2:

```
clear all; close all;clc
[y,Fs] = audioread('music1.wav');
tr_piano=length(y)/Fs;  % record time in seconds

n = length(y);
t = (1:n)/Fs;
y = y(1:n)';
L = t(end);
k = 1/L*[0:n/2-1 -n/2:-1];
ks = fftshift(k);

a = 25;
tslide=0:0.1:L;
ygt_spec = zeros(length(tslide),n);
for j=1:length(tslide)
    g=exp(-a*(t-tslide(j)).^2);
    yg=g.*y;
    ygt=fft(yg);
    ygt_spec(j,:) = fftshift(abs(ygt)); % We don't want to scale it
end

figure(1)
pcolor(tslide,ks,ygt_spec.'),
shading interp
xlabel('Time (s))'), ylabel('Frequency(Hz)')
set(gca,'Ylim',[-1000 1000],'Fontsize',16)
colormap(hot)
%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(2)
[y,Fs] = audioread('music2.wav');
tr_rec=length(y)/Fs;  % record time in seconds
n = length(y);
t = (1:n)/Fs;
y = y(1:n)';
L = t(end);
k = 1/L*[0:n/2-1 -n/2:-1];
ks = fftshift(k);

a = 25;
tslide=0:0.1:L;
ygt_spec = zeros(length(tslide),n);
```

```
for j=1:length(tslide)
    g=exp(-a*(t-tslide(j)).^2);
    yg=g.*y;
    ygt=fft(yg);
    ygt_spec(j,:) = fftshift(abs(ygt)); % We don't want to scale it
end

figure(1)
pcolor(tslide,ks,ygt_spec.'),
shading interp
set(gca,'Ylim',[-2000 2000],'Fontsize',16)
colormap(hot)
%plot((1:length(y))/Fs,y);
%xlabel('Time [sec]'); ylabel('Amplitude');
%title('Mary had a little lamb (recorder)');
%p8 = audioplayer(y,Fs); playblocking(p8);
```