HW 2: Gábor Transforms

Anamol Pundle

February 5, 2015

Abstract

The process and results of generating a spectrogram from audio signals using a windowed Fourier transform(Gábor transform) is described in this report. Three audio signals of varying length and sampling rate are considered. A time-filtering window is constructed and translated across the audio signals. A spectrogram is then generated from the Fourier transform of each snapshot taken by the time filtering window. Several types of functions are used as the time-filtering window, and the results from each are analyzed. The effect of several other parameters, such as the window width and time step of the window (oversampling vs. undersampling) on the spectrogram are also explored.

1 Introduction and Overview

Time frequency analysis comprises of techniques that simultaneously study a signal in time domain and frequency domain. This is useful for analyzing audio signals, since they include several frequencies (and their overtones) played at different moments in time. In this work, three audio signals are analyzed using time frequency analysis and their spectrograms are created. The audio signals are the Hallelujah chorus from Handel's 'Messiah', and renderings of the popular nursery rhyme 'Mary had a Little Lamb' on piano and recorder.

Spectrograms of the audio signals are created by implementing the Gábor transform, using different functions as the sliding window. The effect of various parameters, such as the window width, oversampling, undersampling and different functions on the spectrogram is explored. The musical score for 'Mary had a Little Lamb' is also generated using the Gábor transform. The differences between the spectral signature of the piano and the recorder is analyzed by studying the spectrogram of the two versions of 'Mary had a Little Lamb'.

2 Theoretical Background

2.1 Time Frequency Analysis and Windowed Fourier Transforms

The Fourier transform is one of the most important tools in signal processing, but it has severe limitations. While the Fourier transform captures all the frequency information of the signal, it fails to capture the moment in time at which the various frequencies occur. The Fourier transform is good for characterizing signals whose frequency does not change with time, but does not work for signals with changing frequencies. Therefore, we need to modify this method in order to analyze signals with changing frequencies in order to extract both time and frequency information. Such an analysis, which extracts both time and frequency information with a loss of resolution in both domains, is called time frequency analysis.

The simplest method of time frequency analysis is to decompose the signal over the time domain into separate time frames. For each time window, the Fourier transform is applied in order to characterize the frequencies present during that time frame. A slightly more complicated method is to have the window translate across the time domain, in order to pick up the frequency data at each instant in time. This is known as the Gábor transform. Several other windowed Fourier transforms exist, such as the Zak transform and the Wigner-Ville distribution.

2.2 Gábor Transform

The Gábor transform, also known as the short term Fourier transform, is defined as

$$G[f](t,\omega) = \int_{-\infty}^{\infty} f(\tau)\overline{g}(\tau - t)e^{-i\omega\tau}d\tau \tag{1}$$

where $g_{(t,\omega)}(\tau) = e^{i\omega\tau}g(\tau - t)$ and the bar denotes the complex conjugate of the function.

The function $\overline{g}(\tau - t)$ acts as a time filter for localizing the signal over a specific time window. The integration over the parameter τ slides the time-filtering window down the entire signal in order to pick out the frequency information at each instant of time. τ is the time at which the function is centered, and a is the width of the function. Figure 1 is a cartoon representation of the fundamental ideas behind a time series analysis, Fourier transform analysis and Ǵabor transform analysis of a given signal. In the time series method, good resolution is achieved of the signal in the time domain, but no frequency resolution is achieved. In Fourier analysis, the frequency domain is well resolved at the expense of losing all time resolution. The Ǵabor method, or short-time Fourier transform, trades away some measure of accuracy in both the time and frequency domains in order to give both time and frequency resolution simultaneously.

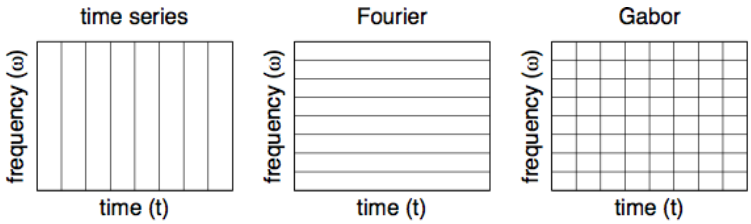


Figure 1: Graphical depiction of the difference between a time series analysis, Fourier analysis and Ǵabor analysis of a signal.

In practice, the Gábor transform is computed by discretizing the time and frequency domain. Thus a discrete version of the transform needs to be considered. Essentially, by discretizing, the transform is done on a lattice of time and frequency. Thus consider the lattice, or sample points,

$$\nu = m\omega_0$$

$$\tau = nt_0$$

where m and n are integers and $\omega_0, t_0 > 0$. Then the discrete version of $g_{t,w}$ becomes

$$g_{m,n}(t) = e^{i2\pi\omega_0 t} g(t - nt_0) \tag{2}$$

If $0 < t_0, \omega_0 < 1$, then the signal is over-sampled and time frames exist which yield excellent localization of the signal in both time and frequency. If $\omega_0, t_0 > 1$, the signal is under-sampled and the Gábor lattice is incapable of reproducing the signal.

Several functions can be used for the Gábor window; one of the simplest is the Gaussian function, given by

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

where a and b are variables that describe the filter width and the center of the function, respectively. If the exponent to which $(t - b)$ is raised to is increased, we get a hyper Gaussian.

Another function for the Gábor window explored in this work is the Mexican hat wavelet, so named due to it's resemblance to a Sombrero. The Mexican hat function is the second derivative of the Gaussian wave and (for application in this work) is defined as

$$g(t) = (1 - bt)e^{-a(t-c)^2} \tag{4}$$

where c describes the center of the wave, a describes the width of the function and b describes the maximum negative value reached by the function as well as

it's curvature. Another function explored in this work is the Shannon window, or a square wave. The data gathered by using windowed Fourier transforms can be used to develop a spectrogram.

2.3 Spectrograms

A spectrogram is a visual representation of a spectrum of frequencies in a signal as they vary with time. In this work, spectrograms are constructed using frequency data gathered using the Gábor transform on three audio signals. Spectrograms are useful tools because they provide information about the frequencies present at a moment in time, as opposed to the Fourier transform, which gives only frequency information. The time-frequency analysis can can be used to produce speech recognition algorithms given the characteristic signatures in the time-frequency domains of sounds. Thus spectrograms are a sort of fingerprint of sound. Several examples of spectrograms are shown in Section 4 of this report.

3 Algorithm Implementation and Development

Spectrograms for three audio signals are developed in this work. The effect of the size and type of window used on the spectrogram is also analyzed. The algorithm used for all three spectrograms is quite similar, and is listed below.

- Import .wav file(s)/load audio file into MATLAB. Calculate its length and the sampling rate(number of points).
  - The audio vector needs to be transposed since it is in a column.
- Define grid vectors
  - The grid vectors for the frequency domain need to be multiplied by $\frac{2\pi}{L}$ since the FFT algorithm assumes 2π periodic signals. These grid vectors also need to be shifted using the `fftshift` function, since the FFT algorithm shifts the data such that $x \in$ [-L, 0]$\rightarrow$[0,L] and $x \in [0, \mathrm{L}]\rightarrow$[-L,0].
- Define a time-slide vector for the sliding window (the time by which the window is translated).
- Start loop for translating window.
- Inside the loop, define the function for the translating window g. This is either a Gaussian function, Shannon window or a Mexican hat function.
- Multiply the audio signal by the translating window function, and take the Fourier transform of the resulting function.

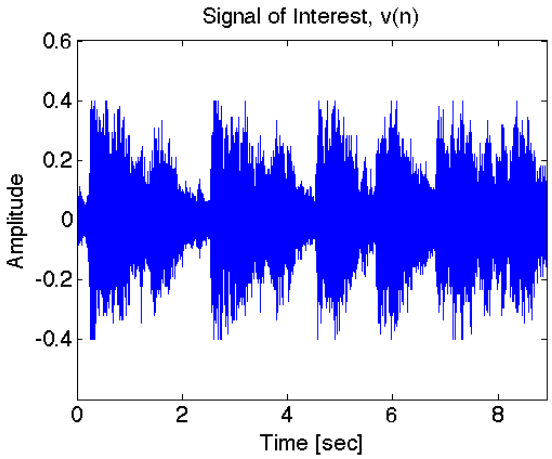


Figure 2: 'Hallelujah' chorus from Handel's 'Messiah'

- Take the absolute and fftshifted value of the resulting resulting vector and add it as a new row to `Sgt_spec`, a matrix from which the spectrogram will be constructed.
- Plot the signal with the translating window, the function resulting from their product and it's Fourier transform.
- Move to next iteration. If loop is over, plot the spectrogram using the `pcolor` function resulting from the `Sgt_spec` matrix with time on the x-axis and frequencies on the y-axis.

The effect of the variables that govern the behavior of window (mentioned in Section 2) is tested heuristically, and the best values for these variables is found.

4 Computational Results

Computational results for the musical piece by Handel and the recordings of a piano and recorder are presented in this section.

4.1 The Handel Piece

This audio signal consists of a piece of music composed by Handel. It runs for 8.93 seconds and is discretized into 73113 points. The sampling rate is 8192 Hz. Figure 2 shows the audio signal in time.

The signal is investigated using the Gábor transform, with a Gaussian function as the sliding window. Figure 3 shows the audio signal with the sliding window on the top, the product of the Gaussian function and the audio signal in the middle, and the Fourier transform of the resulting function at the bottom.

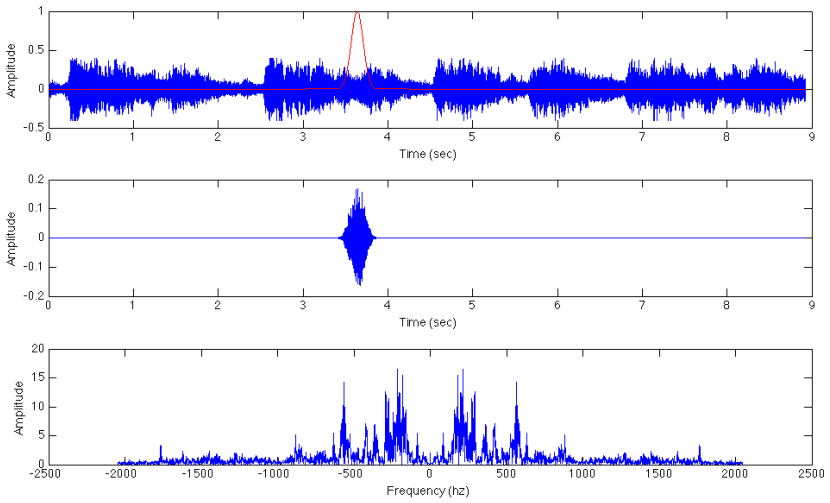


Figure 3: Top: Audio signal with sliding window; Middle: Gaussian function multiplied by audio signal; Bottom: Fourier transform of function

Figure 4 shows a spectrogram of the audio signal, created using a Gaussian function with $a = 500$ and a time step of 0.01 seconds. Therefore, the width of the filter is approximately 0.25 seconds. Since the lowest frequency that can be heard by humans in 20 Hz, and the filter width is greater than 1/20 seconds, all frequencies are captured by the filter. In this case, the signal is oversampled, and hence a good localization of both time and frequency is obtained. Two major groups of frequencies can be isolated; one in the 500 - 600 Hz range and the other in the 260 - 300 Hz range, which appear to be the 'Hallelujah' chorus sung by women (higher frequencies) and men (lower frequencies). Other frequencies are not shown well in this plot because the amplitude of the other frequencies is lower than the amplitude of the frequencies mentioned above.

Figure 5 shows a comparison of the spectrograms generated by increasing the size of the Gábor window from approximately 0.25 seconds to 1.5 seconds, or varying $a$from 500 to 1. The time step of the sliding window is kept constant at 0.01 seconds. Figure 5(a) shows a good resolution in time as well as space. As mentioned before, all frequencies are caught by the filter, since the filter width is much larger than the largest wavelength of any frequency. The resolution becomes a little worse in Figure 5(b), and very poor in Figure 5(d).

Figure 6 shows the effect of undersampling and oversampling the audio signal. An oversampled signal gives good localization in time and space. An undersampled signal cannot fully recreate the signal, since all the data is never sampled. Figure 6(a) shows an undersampled spectrogram, with a filter width of 0.25 seconds and a time step of 0.6 seconds. As can be seen, several frequencies

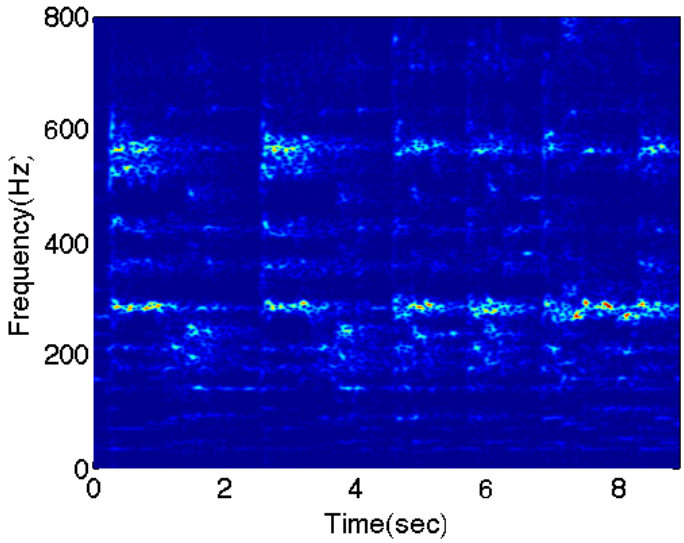
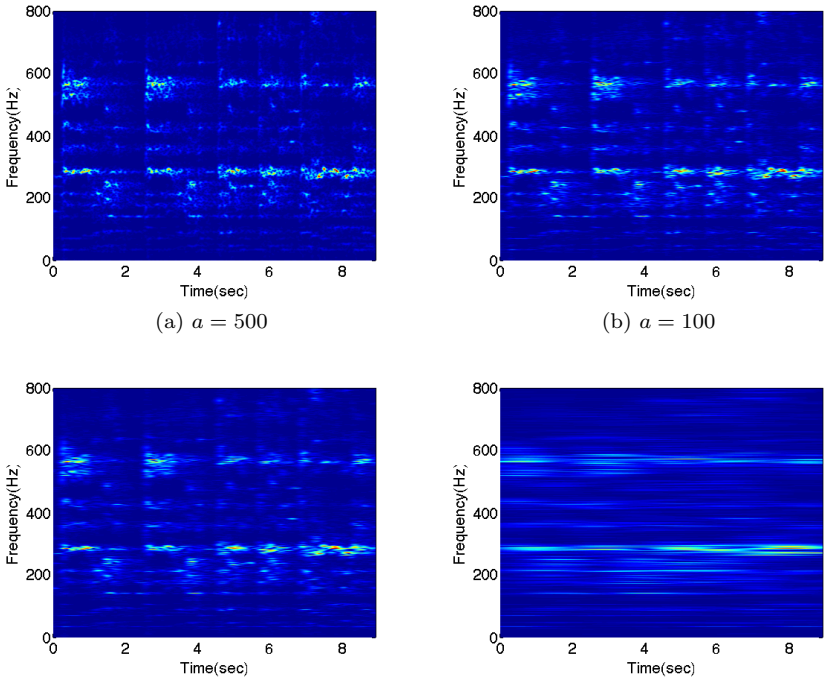


Figure 4: Spectrogram of Handel's 'Messiah'. $a = 500, \Delta t = 0.01s$



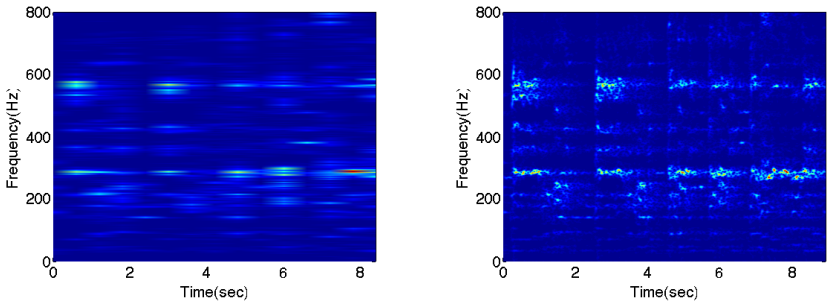
(a) $a = 500$

(b) $a = 100$

(c) $a = 50$

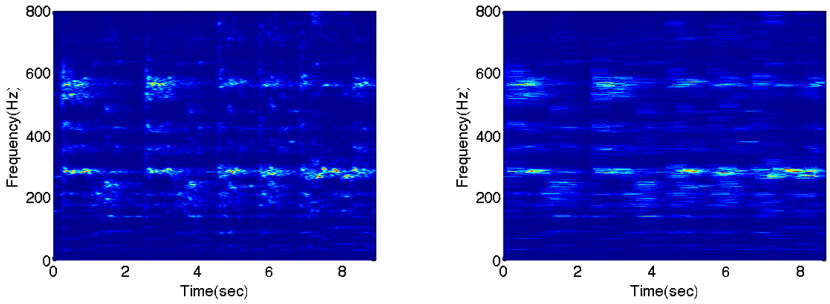
(d) $a = 1$

Figure 5: Spectrograms showing increasing size of window for $\Delta t = 0.01$ seconds



(a) Undersampled spectrogram (b) Oversampled spectrogram

Figure 6: Effect of undersampling and oversampling on spectrogram



(a) Mexican hat function (b) Shannon window

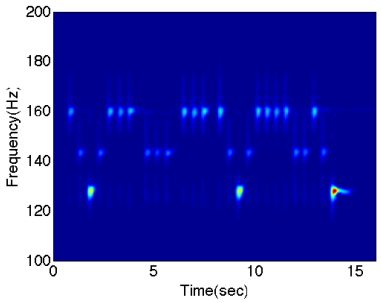
Figure 7: Spectrograms generated by the Mexican hat function and the Shannon window

are missing from the spectrogram, since they have not been sampled. Figure 6(b) shows an oversampled spectrogram, the same as Figure 2.

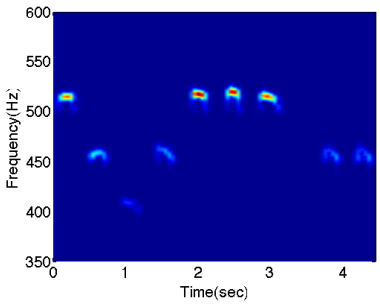
Figure 7a shows spectrogram developed using and Mexican hat function. The time step of the sliding window for the Mexican hat is 0.01, and the window width is about 0.5 sec. The Mexican hat function generates a good spectrogram, quite similar to that generated by the Gaussian function. The spectrogram generated by the Shannon window, however, is seen to be shaped into 'blocks', and frequencies are not as well resolved as the Gauss function and the Mexican hat function.

4.2 Mary Had a Little Lamb

The section of the report analyzes two versions of 'Mary had a Little Lamb'. One is played with a piano and the other is played with a recorder. The piano recording comprises of 701,440 points and is 16 seconds long, while the recorder



(a) Spectrogram of piano



(b) Spectrogram of recorder. Only 4.5 seconds could be analyzed due to computational constraints

Figure 8: Spectrogram of 'Mary had a Little Lamb' played using piano and recorder

piece is 14 seconds long and is discretized into 627,712 points.

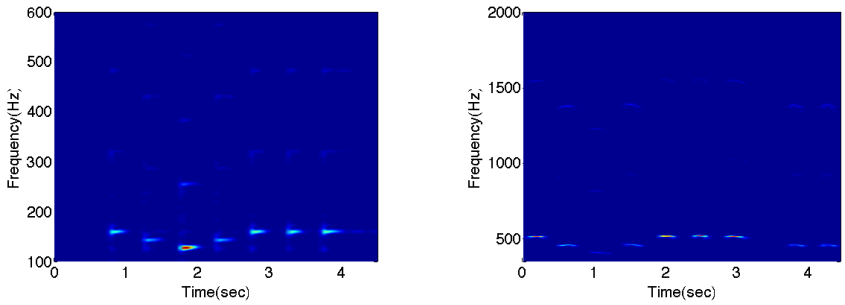
Both pieces are analyzed using the Gabor transform, with the Gaussian function as the sliding window. Only a quarter of the recorder's signal was analyzed due to the large computational and memory requirements. The window width is taken to be 0.5 sec, and the time step is taken to be 0.05 sec. Figure 8 shows the spectrograms of both the audio signals, with overtones removed. Overtones are integral multiples of the fundamental frequency, which give the instrument its 'timbre'. The piano signal has frequencies ranging from 120 Hz to 180 Hz, while the recorder has frequencies ranging from 400 Hz to 525 Hz. This is consistent with the graph on musical instruments given in the HW problem statement. Only a quarter of the recorder's signal was analyzed due to the large computational and memory requirements.

Figure 9 shows the spectrogram of the piano and recorder, this time including overtones. Though the overtones are lower in amplitude than the fundamental frequency, they are responsible for the way an instrument sounds.

5 Summary and Results

Three audio signals were analyzed using time frequency analysis. The Gabor transform was successfully used to create spectrograms of the three audio signals. Several functions were used for the sliding window in the Gabor transform for the first audio signal, Handel's 'Messiah'. The effect of increasing the window width and window time step on the spectrogram were studied. It was found that a window width of approximately 0.3 seconds and a window time step of 0.01 seconds is sufficient to generate good localization of time and frequency. Undersampling and oversampling the data were also explored.

The difference between the piano and recorder was brought out by the time



(a) Piano (b) Recorder

Figure 9: Spectrograms showing overtones of piano and recorder

frequency analysis of the other two audio signals, versions of 'Mary had a Little Lamb' using piano and recorder. The frequencies of the piano in this recording ranged from 120 Hz to 180 Hz, while that from the recorder ranged from 400 Hz to 525 Hz. Differences in the overtones of both instruments was also noted. Using the Gabor transform, the musical score of the tune was successfully recreated.

References

1. Kutz, J. Nathan, Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data, September 2013

2. Matlab help

Appendix I: MATLAB Functions Used

`wavread`: reads wave file into MATLAB

- `y = wavread(filename)` loads a WAVE file specified by the string filename, returning the sampled data in y. If filename does not include an extension, `wavread` appends .wav

`length`: length of largest array dimension

- `L = length(X)` returns the length of the largest array dimension in X. For vectors, the length is simply the number of elements. For arrays with more dimensions, the length is `max(size(X))`.

`plot`: creates a 2-d line plot.

- `plot(X,Y)` creates a 2-D line plot of the data in Y versus the corresponding values in X.

`fft`: One dimensional discrete Fourier transform

- `Y = fft(x)` returns the discrete Fourier transform (DFT) of vector x, computed with a fast Fourier transform (FFT) algorithm.

`pcolor`: creates a pseudocolor (checkerboard) plot

- `pcolor(C)` draws a pseudocolor plot. The elements of C are linearly mapped to an index into the current colormap. The mapping from C to the current colormap is defined by colormap

Appendix II: MATLAB Code

Matlab code for Handel's 'Messiah'

```
clear all; close all; clc;
load handel %load audio file
a = 10; %set window width for Gaussian wave/Mexican hat
ShaWid = 0.25; %set window width for Shannon window
b = 100;

v = y'/2;
L = length(v)/Fs;
k=(2*pi/(2*L))*[0:(length(v)-1)/2 -(length(v)-1)/2:-1]; ks=fftshift(k);

tfinal = length(v)/Fs; %Final time of signal
t = (1:length(v))/Fs;

Sgt_spec = []; tslide = 0:0.01:tfinal; %set sliding window time step

figure(2)
for ii = 1:length(tslide)
     g = exp(-a*(t-tslide(ii)).^10); %Gaussian window
     %Shannon window
%     if tslide(ii) + ShaWid > 8.92 % break;
%     end
%     g = 0 * [1:length(v)];
%     if tslide(ii) < 2 * ShaWid
%          tend = floor((tslide(ii) + ShaWid) * 8192);
%          g(1:tend) = 1;
%     elseif tfinal - tslide(ii) < 2 * ShaWid
%          tbeg = ceil((tslide(ii) * 8192)) + 1;
%          g(tbeg:end) = 1;
%     else
%          tbeg = floor((tslide(ii) - ShaWid) * 8192);
%     tend = floor((tslide(ii) + ShaWid) * 8192);
%          g(tbeg:tend) = 1;
```

```
%     end
%Mexican hat window
%g = (1 - b * (t - tslide(ii)).^2) .* exp(-a*(t-tslide(ii)).^2);

Sg = g.*v; Sgt = fft(Sg); %Multiply window with signal
Sgt_spec = [Sgt_spec; abs(fftshift(Sgt))]; %Take FFT of result
%Plots
subplot(3,1,1), plot((1:length(v))/Fs,v, (1:length(v))/Fs, g, 'r');

xlabel('Time (sec)'); ylabel('Amplitude');
set(gca, 'Fontsize', 14);
subplot(3,1,2), plot((1:length(v))/Fs, Sg);
xlabel('Time (sec)'); ylabel('Amplitude');
set(gca, 'Fontsize', 14);
subplot(3,1,3), plot(ks/(2*pi), abs(fftshift(Sgt)));
xlabel('Frequency (hz)'); ylabel('Amplitude');
set(gca, 'Fontsize', 14); pause(0.3)
end

%Create spectrogram
figure(5)
pcolor(tslide,ks/(2*pi),Sgt_spec.'), shading interp
set(gca,'Ylim',[0 800],'Fontsize',[20])
xlabel('Time(sec)'); ylabel('Frequency(Hz)');
```

Matlab code for 'Mary had a little lamb'

```
clear all; close all; clc;
tr_piano=16; % record time in seconds
y=wavread('music1'); %load audio file
Fs=length(y)/tr_piano;

v = y'/2;
a = 500; %set window width for Gaussian function
L = length(v)/Fs;

k=(2*pi/(2*L))*[0:length(v)/2-1 -length(v)/2:-1]; ks=fftshift(k);

tfinal = length(v)/Fs; %calculate final time of signal
t = (1:length(v))/Fs;

Sgt_spec = [];
tslide = 0:0.05:tfinal; %vector for time step of sliding window
figure(2)
```

```
   for ii = 1:length(tslide)
        g = exp(-a*(t-tslide(ii)).^2); %Gaussian window
        Sg = g.*v; Sgt = fft(Sg); %multiply window with signal and
take fft
        Sgt_spec = [Sgt_spec; abs(fftshift(Sgt))]; %add to matrix
        %plots
        subplot(3,1,1),plot((1:length(v))/Fs,v,(1:length(v))/Fs,g,'r');

        axis([0 length(v)/Fs -0.5 1]);
        subplot(3,1,2), plot((1:length(v))/Fs, Sg);
        axis([0 length(v)/Fs -0.5 1]);
        subplot(3,1,3), plot(ks, abs(fftshift(Sgt)));
        pause(0.05)
   end

   %Create spectrogram
   figure(5)
   pcolor(tslide,ks/(2*pi),Sgt_spec.'), shading interp
   set(gca,'Ylim', [100 600],'Fontsize',[14])
```