Gabor Transforms

AMATH 482 Homework 2

Robin Yan

Feb. 2, 2020

Abstract

Gabor transform, or short time Fourier transform, provides a new method to handle data, time-frequency analysis. Compared with Fourier analysis, Gabor can analyze both time and frequency domain. This homework demonstrates the capability of Gabor transform and its various applications in data analysis. This assignment consists of two parts, the first being an experimental survey of the different parameters in Gabor transform and their effects on resulting spectrograms, with a sample data, and the second being an application of Gabor in comparative data analysis with actual acquired data. The script uses Gabor transform to generate spectrograms of a preloaded sound data, identify the music scores of two recorded music pieces and characterize the overtones of different instruments from the two pieces.

Section I – Introduction and Overview

The purpose of this homework is to explore the characteristics and capabilities of Gabor transform, which are divided into two parts with separate data. The first part involves analyzing a portion of Handel's Messiah that is embedded in MATLAB with spectrograms. The goal of this part is to tune the window width, sampling rate and Gabor window to see their respective effects on the final spectrograms.

The second part focuses on analyzing two pieces of song "Mary had a little lamb" played by two instruments, piano and recorder. The goal of this part is to identify the music scores of the data through Gabor transform and distinguish the characteristics of the two instruments by analyzing their overtones.

Section II – Theoretical Background

Fourier Transform

Fourier transform and its inverse, elaborated in Report 1, are defined as:

$$F(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ikx} f(x) dx \quad (E.1)$$

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{ikx} F(k) dk \quad (E.2)$$

In practice, the computational domain is limited to a finite domain from $-\frac{L}{2}$ to $\frac{L}{2}$, and carried out with fast Fourier transform for higher efficiency.

Gabor Transform

Gabor transform uses a filter, called Gabor window, to selectively perform Fourier transform on a small segment of the original data. The process is repeated by shifting the Gabor window along the time domain to acquire frequency information in known time windows.

Gaussian Filter

Gaussian filter is one of the common Gabor windows in use, and is written as:

$$\mathcal{F}(t) = e^{-a(t-t_{slide})^2} \quad (E.3)$$

In which a is the bandwidth coefficient of the filter, positively related to the decay rate of the filter and t_{slide} is the central location of Gabor window.

Mexican Hat Wavelet

Mexican hat wavelet is usually used in wavelet analysis, but can also serve as a signal filter in Gabor transform, and is written as:

$$\mathcal{F}(t) = (1 - (t - t_{slide})^2)e^{-\frac{(t-t_{slide})^2}{2}} \quad (E.4)$$

Shannon Window (Step-function)

Step function can also be used to filter data in Gabor transform, and is defined in MATLAB as:

$$\mathcal{F}(t) = 1 \quad if \quad abs(t - t_{slide}) \leq \frac{a}{2} \quad (E.5)$$

In which a is the width of the function, and the function equals zero elsewhere.

Section III – Algorithm Implementation and Development

Data Preparation

The initialization of the two parts is largely similar to the previous project, which involves loading sound data into a vector and sampling frequency into a scaler, computing time domain labels based on sampling frequency, creating frequency domain labels by scaling MATLAB default domain of $2\ \pi$ to the length of time domain L, and shifting frequency labels for easier plotting. Part one requires special indexing since the sampling point is odd, see Appendix B.

Part I - Basic Spectrogram

The most basic spectrogram is generated by applying a Gaussian filter (E.3) to the music piece and performing Fourier transform (E.1) to compute the frequency information in the window. The filter center, t_{slide}, is sequentially shifted by unit time step, dt, followed with another Fourier transform at the new window. With frequency label converted to Hz forming the y axis, the final spectrogram is a colormap showing frequency signature at varying time point throughout the data. For this basic setup, the window width and time step are arbitrarily determined to be 1 and 0.1s.

Part I – Different Window Width

For this section, the Gaussian bandwidth coefficient a, inversely related to the width of the filter, varies from 1 to 1000, with tenfold increment. The time step is held constant as 0.1s. Subsequent spectrograms are generated to compare the effects resulting from different window width.

Part I – Different Time Step

In this section, the time step for t_{slide}, dt, varies from 1s to 0.01s, with tenfold increment. The window width is set to 100 based on observations from previous section. Subsequent spectrograms are generated to compare the effects resulting from different time step.

Part I – Different Gabor Window

This section focuses on using different filters for Gabor transform. The filters include Gaussian, Mexican hat wavelet (E.4) and step-function (E.5). The widths of each filter are set to ensure the most similar coverage across all three filters. The time step is held constant at 0.1s.

Part II – Music Scores Identification

To ensure efficient identification, the width of the filter is iterated to capture the most information from each note without receiving too much interference from neighboring notes. The final Gaussian coefficient, a, is determined to be 50.

Instead of progressing through time with constant time step, the filter centers, t_{slide}, are specified by finding the local maximum points in the time domain, corresponding to the keystrokes in the music. The minimum separation distance is set to 0.3 to ensure points can represent the distinct keystrokes. The minimum prominence is adjusted to 0.4 for piano and 0.1 for recorder to reduce interference from high frequency sound.

Spectrograms are generated with the same protocol as Part I, with frequency in Hz and time in s. The frequencies with maximum amplitude at all filter centers (keystrokes) are determined and exported to a separated Excel sheet. The two lists of frequencies, in Hz, from both piano and recorder, are manually identified and mapped into music scores based on the given conversion sheet.

The spectrograms generated in this part are validated by spectrograms from MATLAB's built-in function.

Part II – Overtone Comparison

Due to their higher contrasts, the spectrograms generated with MATLAB's built-in command, consistent with explicitly generated spectrograms, are used for analysis in this section.

To compare the overtone signature of the two instruments, the script zooms in at the first note of the piece in each spectrogram for the best horizontal comparison. Gabor window size, overlapping length and sampling points are specified to 1000, 500 and 2^{12} to ensure smoothness.

The two magnified spectrograms are cross-referenced to qualitatively compare the characteristics of the piano and recorder.

Section IV – Computational Results and Discussions

Part I - Basic Spectrogram

The basic spectrogram from a portion of Handel's Messiah is shown below in Fig. 1. Window width is 1 and time step is 0.1s. This diagram serves as a baseline for all following spectrogram comparison.

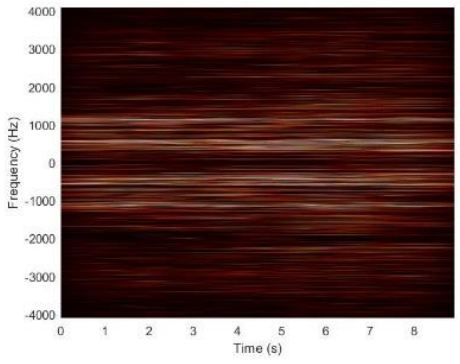


Fig. 1 Basic spectrogram of Messiah.

Part I – Different Window Width

Spectrograms with Gaussian bandwidth coefficient 1, 10, 100 and 1000, time step constant at 0.1s are shown below in Fig. 2.

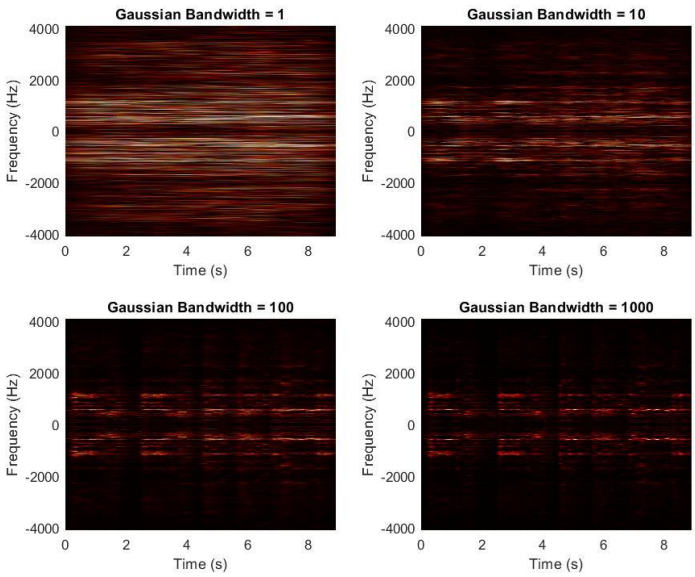


Fig 2. Spectrograms for window width comparison

From the comparison, at 1, because of the large window size, the Gabor transform is largely similar to Fourier transform, with high resolution in the frequency domain but less information in time domain. At bandwidth 10, more time information emerges at the pauses between the vocals are discernible. Further increased to 100, time information is significantly improved, with clearly identifiable vocal sections while frequency information is legible. But at 1000, most of the frequency information is gone due to merging between neighboring frequencies while time resolution further improves.

Based on the results, window width corresponding to bandwidth coefficient 100 is chosen for following analysis.

Part I – Different Time Step

Spectrograms with time steps 1s, 0.1s and 0.01s, window width constant at 100 are shown below in Fig. 3. Because of the significantly different computation time, elapse time is also included in the results for comparison and optimization.

While time step of 1s offers very coarse spectrogram image, time steps of both 0.1s and 0.01s present smooth and informational spectrograms. However, further decrease in time step comes at the expense of more computation time. Recommendation of time step will depend on the type of data, desired resolution and computation power available.

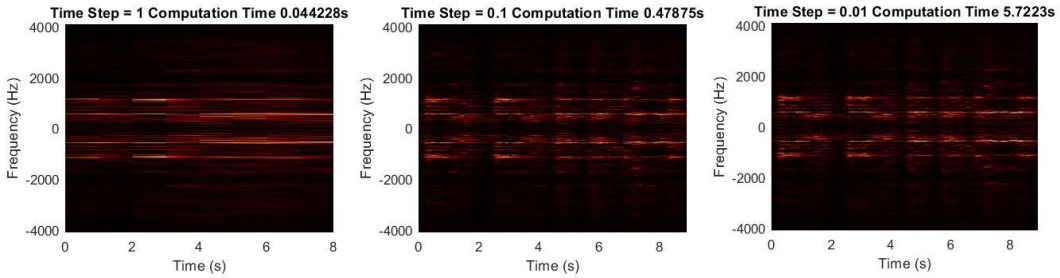


Fig 3. Spectrograms for time step comparison

Part I – Different Gabor Window

Spectrograms with three different filters, Gaussian, Mexican hat and Shannon, with their corresponding filter window at last filter center are shown below in Fig. 4.

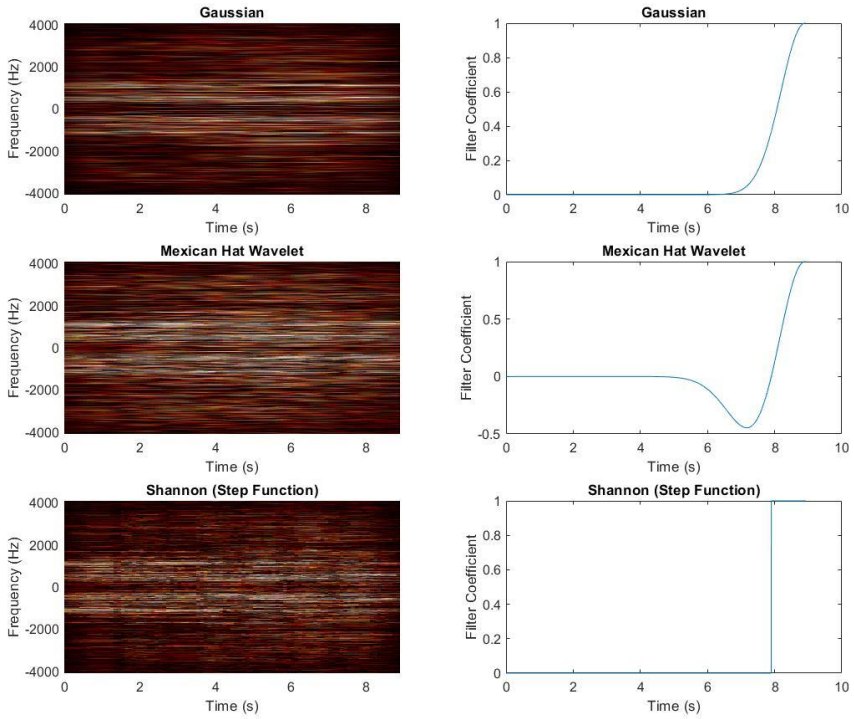


Fig 4. Spectrograms for Gabor window comparison

Again, the three filters are adjusted to ensure controlled coverage. Using Gaussian as the baseline, Mexican hat wavelet shows more noise with similar window size, possibly due to the negative filter coefficient near the window boundary. Step function has coarser spectrogram with similar window size, potentially due to the crisp boundary of the filter, which results in more frequencies to model the sharp change in time domain. Thus, the recommendation for Gabor window depends on the data. In this analysis, Gaussian provides the best results.

Part II – Music Scores Identification

The spectrogram of the piano version of “Mary has a little lamb” is shown here in Fig. 5 to illustrate the music in time-frequency analysis. Each frequency highlight corresponds to a keystroke at that specific frequency in Hz.

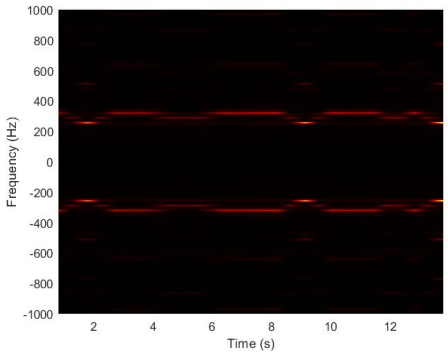


Fig. 5 Spectrogram of piano version

Spectrograms for recorder and those generated by built-in functions are attached in Appendix C as Fig. C4 to C6. Keystroke identification are shown in Appendix C Fig C7 and C8. The translated music scores are attached in Appendix D.

Part II – Overtone Comparison

The spectrograms of the first note in both instruments are shown below in Fig. 6.

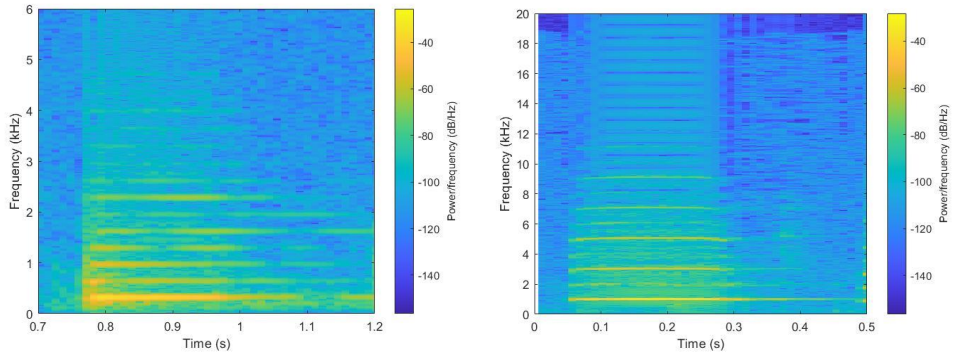


Fig. 6 Magnified Spectrograms (left: piano, right: recorder)

Based on the two spectrograms, piano has base tones with lower frequency and overtones decrease in magnitude away from base. Both base tones and overtones from piano have relatively wide range. Recorder shows base tones with higher frequency and its overtones display oscillating magnitudes with high magnitude overtones alternate with low magnitude ones. Both base tones and overtones are also narrower in frequency.

Section V – Summary and Conclusions

From the results, Gabor transform display various changes when parameter changes. Therefore, it is recommended to tune the parameters based on actual data. It also proves to be useful in analyzing time-sensitive sound data for frequency identification. The script successfully deciphers music scores from two music recordings and shows the difference between different instrument through time-frequency analysis.

In conclusion, coupled with good indexing techniques, Gabor transform can be extremely useful in data analysis.

Appendix A – MATLAB functions used

- fft: Fourier transform
- $fftshift$: shift the zero-frequency component of Fourier coefficients to the center
- $pcolor$: 2D color plot for a matrix of data
- $colormap$: specify the type of color plot
- $tic\ toc$: calculate elapse time
- $audioread$: read audio files into a vector with sampling frequency
- $isolocalmax$: determine local maximum points
- $spectrogram$: generate spectrogram

Appendix B – MATLAB codes


```
clear; close all; clc;
```

Part I - Initialization

```
load handel
v = y';
t = (1:length(v))/Fs;
L = t(end);
n = length(t);
k=(2*pi/L)*[0:(n-1)/2 -(n-1)/2:-1]; % n is odd, translate into omega
ks=fftshift(k);
```

I - Basic Spectrogram

```
a = 1; % window width (inverse)
dt = .1; % time step
tslide=0:dt:L;
vt_spec = zeros(length(tslide),length(v));
for j=1:length(tslide)
    g=exp(-a*(t-tslide(j)).^2);
    vg=g.*v;
    vgt=fft(vg);
    vt_spec(j,:) = fftshift(abs(vgt));
end

figure(1)
pcolor(tslide,(ks./(2*pi)),vt_spec.') % k in terms of Hz
ylabel('Frequency (Hz)')
xlabel('Time (s)')
shading interp
colormap(hot)
```

I - Different Window Width

```
figure(2)
for jj = 1:4
    a = 1*10.^(jj-1); % window width (inverse)
    dt = .1; % time step
    tslide=0:dt:L;
    vt_spec = zeros(length(tslide),length(v));
    for j=1:length(tslide)
        g=exp(-a*(t-tslide(j)).^2);
        vg=g.*v;
        vgt=fft(vg);
        vt_spec(j,:) = fftshift(abs(vgt));
    end
    subplot(2,2,jj);
    pcolor(tslide,(ks./(2*pi)),vt_spec.')
    title(['Gaussian Bandwidth = ' num2str(a)]);
    ylabel('Frequency (Hz)')
    xlabel('Time (s)')
    shading interp
    colormap(hot)
end
```

I - Different Time Step

```
figure(3)
time = zeros(1,3);
for jj = 1:3
    a = 100; % window width (inverse)
    dt = 1*10^(-jj+1); % time step
    tslide=0:dt:L;
    vt_spec = zeros(length(tslide),length(v));
    tic
    for j=1:length(tslide)
        g=exp(-a*(t-tslide(j)).^2);
        vg=g.*v;
        vgt=fft(vg);
        vt_spec(j,:) = fftshift(abs(vgt));
    end
    time(jj) = toc;
    subplot(2,2,jj);
    pcolor(tslide,(ks./(2*pi)),vt_spec.')
    title(['Time Step = ' num2str(dt) ' Computation Time ' num2str(time(jj)) 's']);
    ylabel('Frequency (Hz)')
    xlabel('Time (s)')
    shading interp
    colormap(hot)
end
```

I - Different Gabor Window

```
figure(4)
% widths in three methods are determined to have similar areas
% Gaussian
a = 1; % window width (inverse)
dt = .1; % time step
tslide=0:dt:L;
vt_spec = zeros(length(tslide),length(v));
for j=1:length(tslide)
    g=exp(-a*(t-tslide(j)).^2);
    vg=g.*v;
    vgt=fft(vg);
    vt_spec(j,:) = fftshift(abs(vgt));
end
subplot(3,2,1);
pcolor(tslide,(ks./(2*pi)),vt_spec.')
title('Gaussian');
ylabel('Frequency (Hz)')
xlabel('Time (s)')
shading interp
colormap(hot)
subplot(3,2,2)
plot(t,g)
title('Gaussian');
ylabel('Filter Coefficient')
xlabel('Time (s)')

% Mexican Hat
dt = .1; % time step
tslide=0:dt:L;
vt_spec = zeros(length(tslide),length(v));
for j=1:length(tslide)
    g=(1-(t-tslide(j)).^2).*exp(-(t-tslide(j)).^2/2);
    vg=g.*v;
    vgt=fft(vg);
    vt_spec(j,:) = fftshift(abs(vgt));
end
subplot(3,2,3);
pcolor(tslide,(ks./(2*pi)),vt_spec.')
title('Mexican Hat Wavelet');
ylabel('Frequency (Hz)')
xlabel('Time (s)')
shading interp
colormap(hot)
subplot(3,2,4)
plot(t,g)
title('Mexican Hat Wavelet');
ylabel('Filter Coefficient')
xlabel('Time (s)')

% Shannon
a = 2; % Filter step width
```

```
dt = .1; % time step
tslide=0:dt:L;
vt_spec = zeros(length(tslide),length(v));
for j=1:length(tslide)
    g=(abs(t-tslide(j)) <= a/2);
    vg=g.*v;
    vgt=fft(vg);
    vt_spec(j,:) = fftshift(abs(vgt));
end
subplot(3,2,5);
pcolor(tslide,(ks./(2*pi)),vt_spec.')
title('Shannon (Step Function)');
ylabel('Frequency (Hz)')
xlabel('Time (s)')
shading interp
colormap(hot)
subplot(3,2,6)
plot(t,g)
title('Shannon (Step Function)');
ylabel('Filter Coefficient')
xlabel('Time (s)')
```

Part II - Piano

```
[y,Fs] = audioread('music1.wav');
tr_piano=length(y)/Fs; % record time in seconds
% p8 = audioplayer(y,Fs); playblocking(p8);
v = y';
t = (1:length(v))/Fs;
L = t(end);
n = length(t);
k=(2*pi/L)*[0:(n)/2-1 -(n)/2:-1];
ks=fftshift(k);

a = 50; % window width (inverse)
t_index = islocalmax(v,'MinSeparation',.3*Fs,'MinProminence',.4);
tslide=t(t_index);
vt_spec = zeros(length(tslide),length(v));
for j=1:length(tslide)
    g=exp(-a*(t-tslide(j)).^2);
    vg=g.*v;
    vgt=fft(vg);
    vt_spec(j,:) = fftshift(abs(vgt));
end

figure(5)
plot((1:length(y))/Fs,y,t(t_index),y(t_index),'r*');
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (piano)');

figure(6)
```

```
pcolor(tslide,(ks./(2*pi)),vt_spec.')
ylabel('Frequency (Hz)')
xlabel('Time (s)')
shading interp
set(gca,'Ylim',[-1000 1000])
colormap(hot)

figure(7)
spectrogram(v,1000,[],[],Fs,'yaxis')
set(gca,'Ylim',[0 1]) % spectrogram default in kHz

[~,I] = max(vt_spec,[],2);
Hz = ks./(2*pi);
scores_piano = abs(Hz(I))';

figure(11)
spectrogram(v,1000,500,2^12,Fs,'yaxis')
set(gca,'Ylim',[0 6],'Xlim',[.7 1.2]) % spectrogram default in kHz
```

Part II - Recorder

```
[y,Fs] = audioread('music2.wav');
tr_rec=length(y)/Fs; % record time in seconds
% p8 = audioplayer(y,Fs); playblocking(p8);
v = y';
t = (1:length(v))/Fs;
L = t(end);
n = length(t);
k=(2*pi/L)*[0:(n)/2-1 -(n)/2:-1];
ks=fftshift(k);

a = 50; % window width (inverse)
t_index = islocalmax(v,'MinSeparation',.3*Fs,'MinProminence',.1);
tslide=t(t_index);
vt_spec = zeros(length(tslide),length(v));
for j=1:length(tslide)
    g=exp(-a*(t-tslide(j)).^2);
    vg=g.*v;
    vgt=fft(vg);
    vt_spec(j,:) = fftshift(abs(vgt));
end

figure(8)
plot((1:length(y))/Fs,y,t(t_index),y(t_index),'r*');
xlabel('Time [sec]'); ylabel('Amplitude');
```

```
title('Mary had a little lamb (recorder)');

figure(9)
pcolor(tslide,(ks./(2*pi)),vt_spec.')
ylabel('Frequency (Hz)')
xlabel('Time (s)')
shading interp
set(gca,'Ylim',[-2000 2000])
colormap(hot)

figure(10)
spectrogram(v,1000,[],[],Fs,'yaxis')
set(gca,'Ylim',[0 2]) % spectrogram default in kHz

[~,I] = max(vt_spec,[],2);
Hz = ks./(2*pi);
scores_recorder = abs(Hz(I))';

figure(12)
spectrogram(v,1000,500,2^12,Fs,'yaxis')
set(gca,'Ylim',[0 20],'Xlim',[0 .5]) % spectrogram default in kHz
```

Published with MATLAB® R2019b

Appendix C – Additional Figures

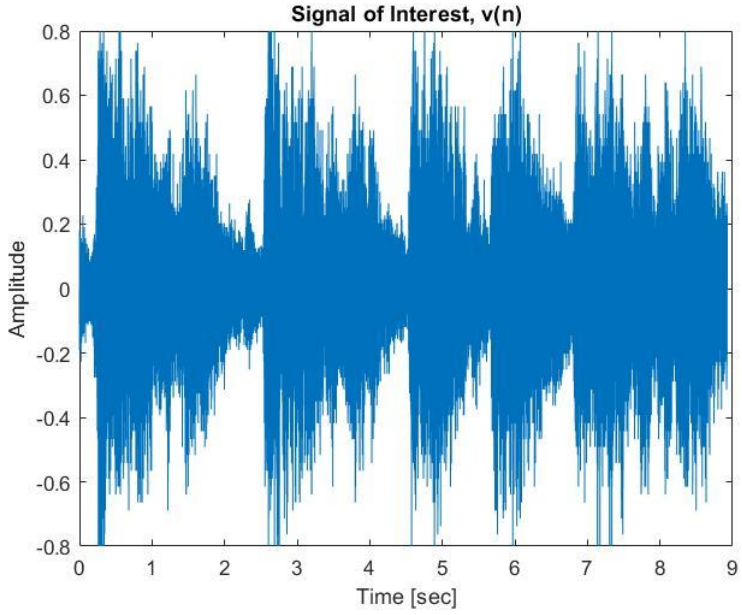
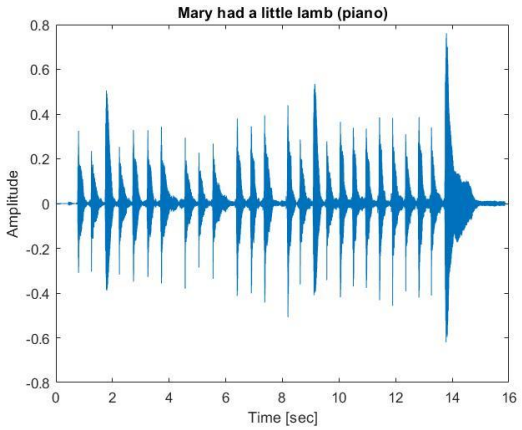


Fig. C1 Raw sound signal of Handel's Messiah



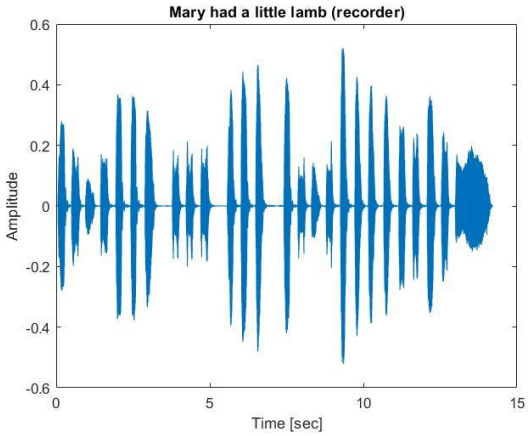


Fig. C2 Raw sound signal of piano version (left)

Fig. C3 Raw sound signal of recorder version (right)

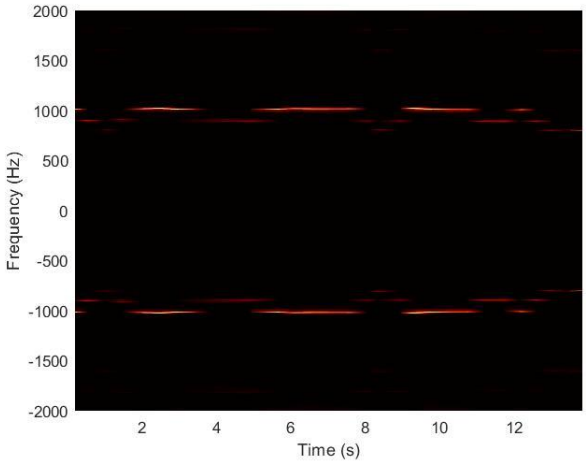


Fig. C4 Spectrogram from recorder version

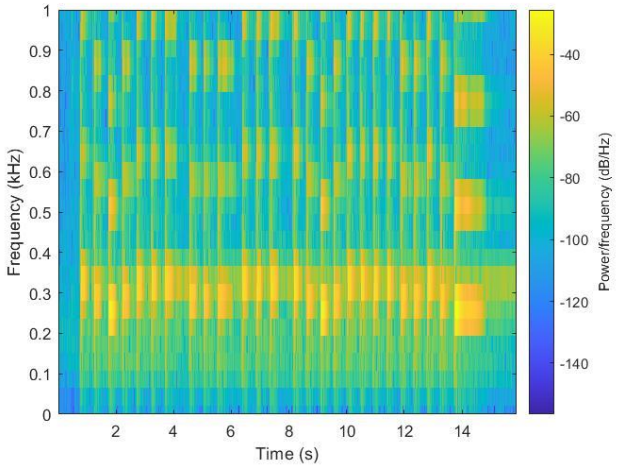


Fig. C5 Spectrogram from piano version from built-in function

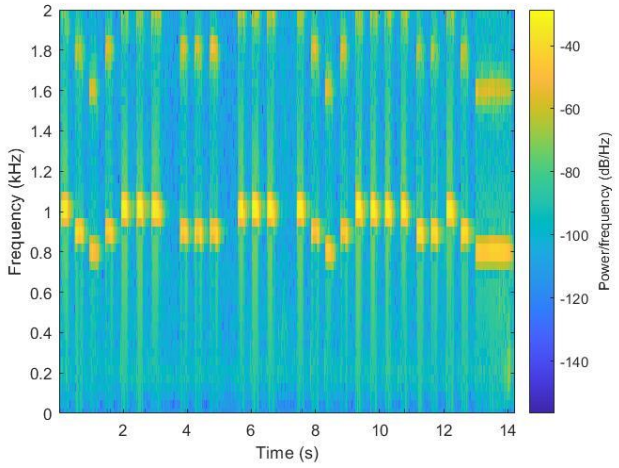


Fig. C6 Spectrogram from recorder version from built-in function

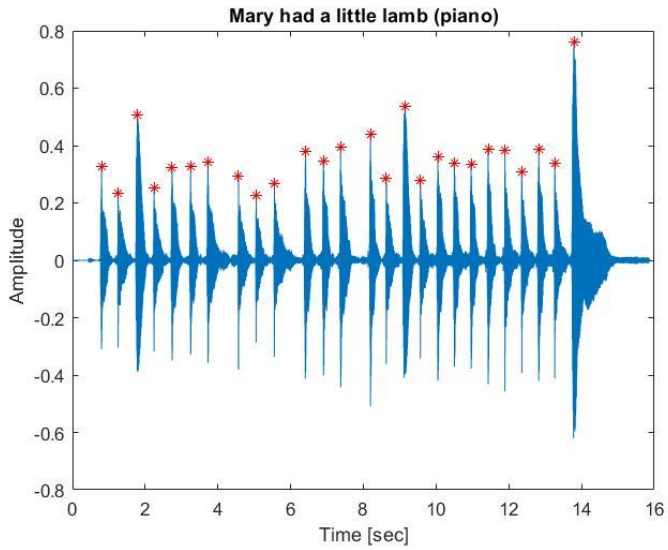


Fig. C7 Piano version with identified keystrokes

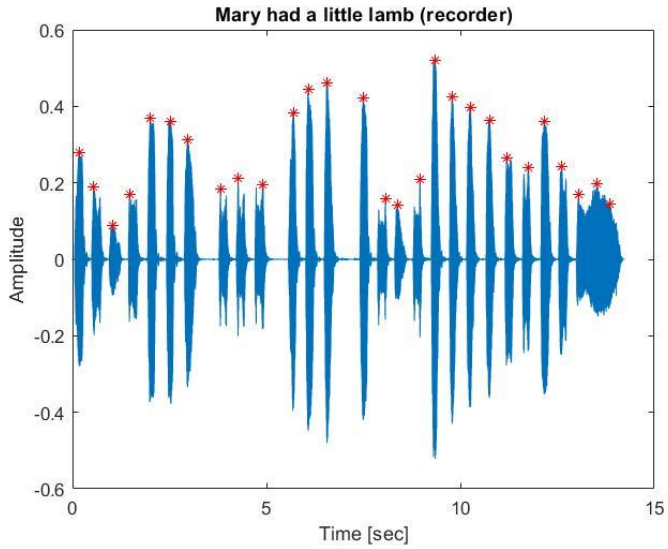


Fig. C8 Recorder version with identified keystrokes (last two discarded in score conversion)

Appendix D – Music Scores

Piano (Hz)			Recorder (Hz)	
321.332	E4		1012.587	B6
288.0105	D4		901.233	A6
257.0153	C4 (middle C)		804.4215	G5
287.9477	D4		908.7504	A6
321.2062	E4		1017.716	B6
321.0176	E4		1021.791	B6
321.8978	E4		1013.15	B6
288.0734	D4		905.2376	A6
288.0734	D4		891.1866	A6
287.8219	D4		893.0132	A6
321.332	E4		1013.29	B6
320.4518	E4		1018.559	B6
321.332	E4		1017.435	B6
321.7092	E4		1017.716	B6
288.1991	D4		896.1044	A6
257.2039	C4 (middle C)		805.405	G5
288.1363	D4		894.2075	A6
321.8349	E4		1022.002	B6
321.4577	E4		1013.852	B6
321.0805	E4		1010.129	B6
322.0864	E4		1009.496	B6
288.3249	D4		895.7531	A6
288.8278	D4		896.5962	A6
321.4577	E4		1008.723	B6
288.6392	D4		895.9639	A6
257.0782	C4 (middle C)		800.3467	A6