Gabor Transforms

Aayush Chhabra

Abstract

We started with the well known classical piece Handel and performed time-frequency analysis on it. We explored the applications of Gabor Transform to perform the analysis and saw the effects of changing various parameters of the gabor window on the corresponding spectrograms. Further, We took a musical piece recorded on both - the piano and the recorder and performed similar analysis on those audio samples. We compared the spectrograms and reproduced their corresponding music scores.

Introduction and Overview

The world, as we observe it, evolves in time and time series data is one of the central topics of study in data analysis. However, there are important insights that can be obtained by looking at the corresponding frequency contents of data. One method of moving from time series data to frequency data is through Fourier Transform. However, the limitation of using Fourier transforms is that there is a complete loss of time resolution in Fourier Transform. To overcome this limitation, one common method is to use Gabor Transforms. In Gabor transforms, a window is used to extract a portion of time series data and then the frequency content of that small portion is analyzed. This allows the possibility of resolving in both time and frequency. In this work, we have used a sliding window to extract data through the entire time scale and looked at its frequency content. We have used various widths of the filter to see the corresponding effects on the final spectrograms. We have also looked at the effects of the stride of the sliding window and compared the case of over sampling with the case of under sampling.

Theoretical Background

The Fourier transform of a function f(t) is defined as $F(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ikx} f(x) dx$. Gabor Transform is just an extension of the Fourier transform which is obtained by changing the Kernel of the Fourier Transform. This new kernel incorporates the action of filtering out a small time slice and allows the combined resolution in both time and frequency.
The Gabor transform of a function f(t) is defined as $G[f](\tau, \omega) = \int_{-\infty}^{\infty} f(t) e^{-\pi(t-\tau)^2} e^{-j\omega t} dt$.

Let's define the generic version of the sliding window for the Gabor Transform: $g = ae^{-b(t-\tau)^2}$ where a and b are used to control the height and the width of the filter. We will use different values of b in our analysis to look at the effects on the corresponding spectrograms.

Algorithm Implementation and Development

PART I

We started by defining the signal for the musical piece handel. Then we defined a gaussian Gabor sliding window with different values of the parameter b to control the width of the filter. Once we defined the filter, we took an element-wise product of the original signal with the Gabor window to obtain a slice of the signal. We used the fft command in Matlab to take this slice of time and obtain the Fourier transform of this slice.

We looped over the entire time axis and created a matrix of frequency values in each time window. Finally, we used the obtained matrix to plot the spectrograms of the data. We studied the data under three different strides of the Gabor window. When the stride was larger (= 1), we sampled data with less frequency and this case was called Under-sampling. When the stride was smaller (= .01), we sampled data with high frequency and this case was called over-sampling. The last case was when the stride was a average number (=0.1) and this case was called the normal case.

PART II

We started with two different .wav files of a song - one recorded on the piano and the other recorded on the recorder. We performed similar analysis of these two files to obtain the corresponding spectrograms. We looked at the maximum value of the signal in the frequency domain to avoid overtones. We also looked at the dominant frequency in each time window and used a frequency scale to map these frequencies to corresponding notes. We combined these notes to obtain music scores.

Computational Results

PART I

In all the cases, we worked with different values of the parameter b. Higher values of the parameter b correspond to narrower Gabor windows and resulted in better time resolution and weaker frequency resolution.

Case 1: Over-sampling - In the case of oversampling, we obtained very accurate results. However, the different between the accuracy of oversampling and normal sampling isn't too much but the computational time increases significantly.

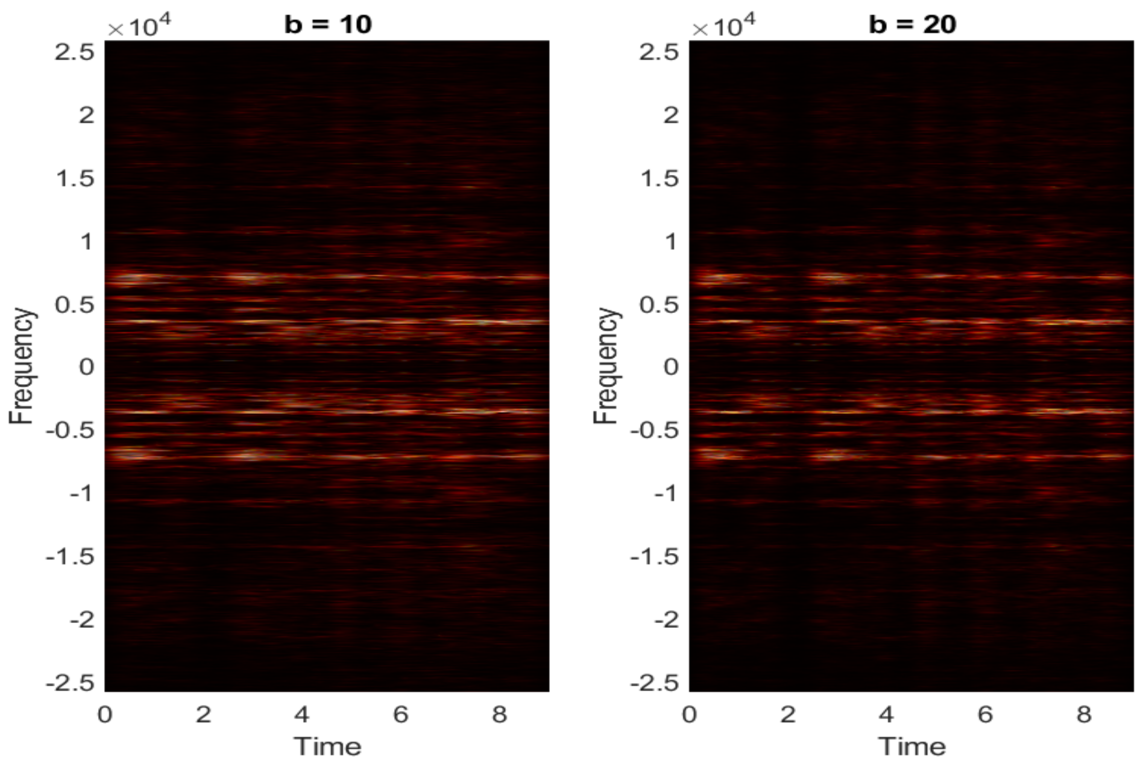

Figure 1: Over-sampling

Case 2: Under-sampling - In the case of under-sampling, we were not able to obtain results with high resolution. The computational time was very low.

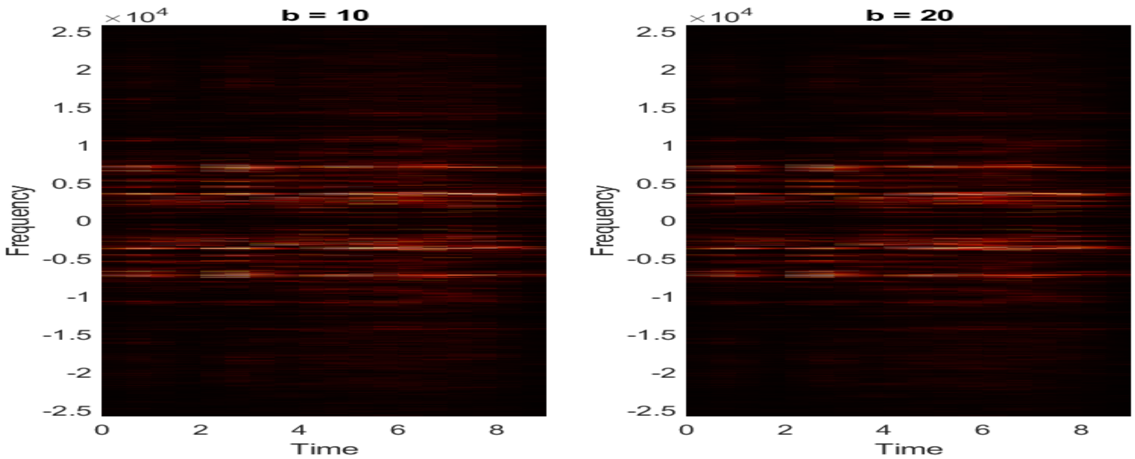

Figure 2: Under-sampling

Case 3: Normal-sampling - We were able to obtain great results with affordable computational times.

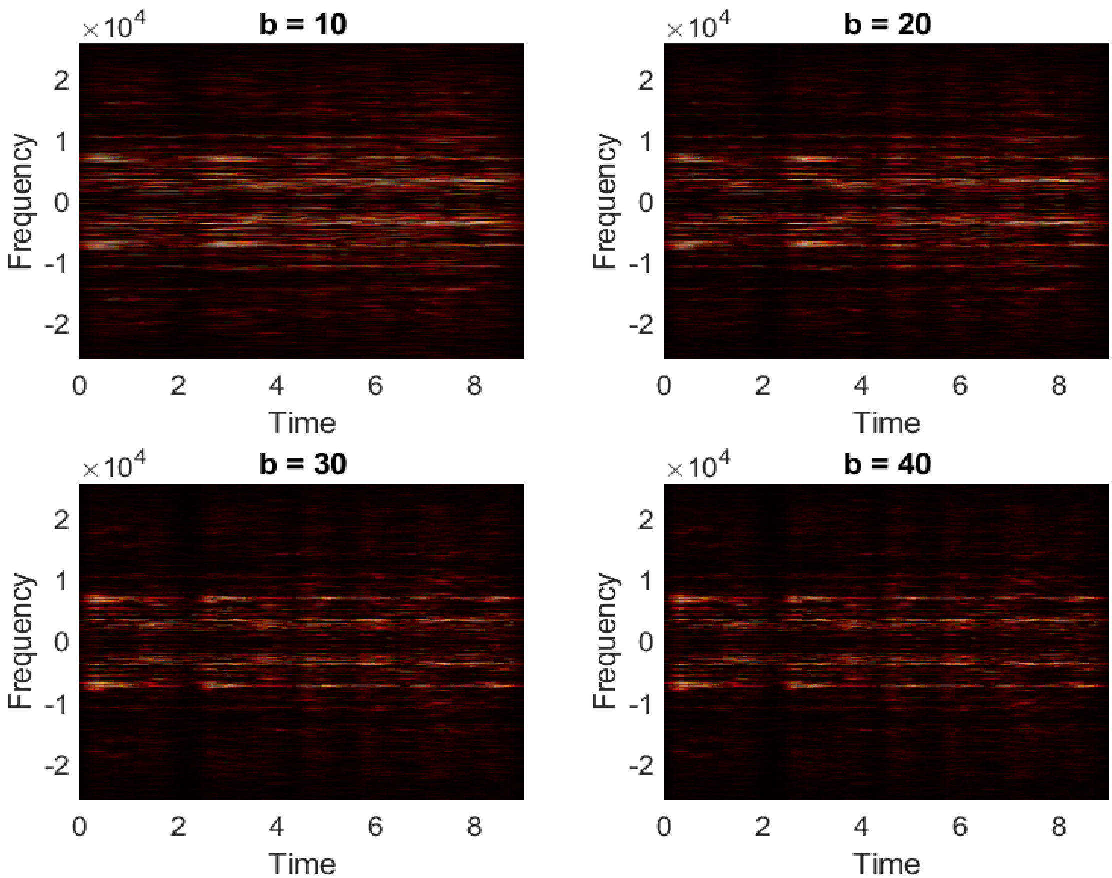

Figure 3: Normal Sampling

PART II

We obtained the spectrograms corresponding to the piano and the recorder audio files. We were also able to map the frequencies from each time window to the corresponding notes to obtain music scores for each instruments. One observation was that although both the instruments had similar general patterns, their operating frequencies were quite different. We also observed that there were higher number of overtones for the piano than there were for the recorder.

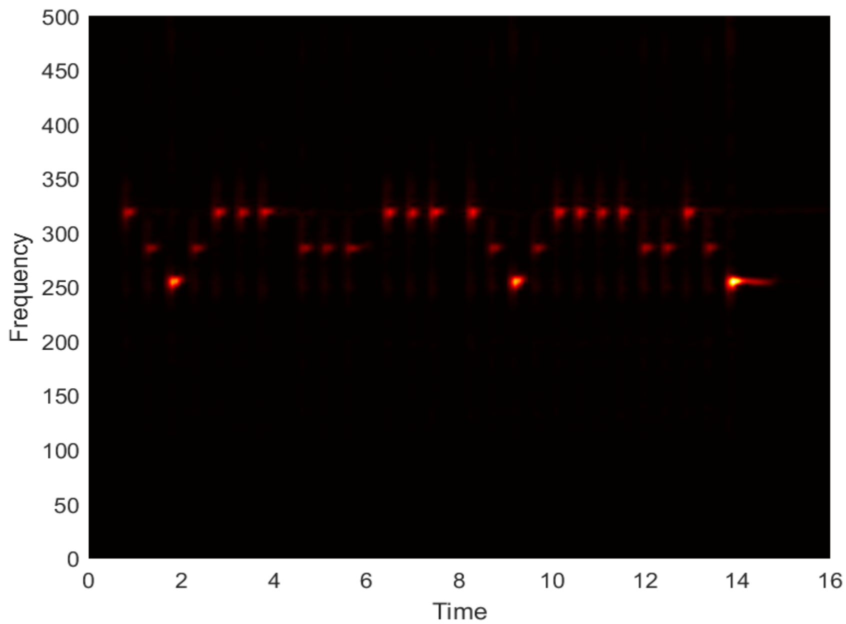

Figure 4: piano spectrogram

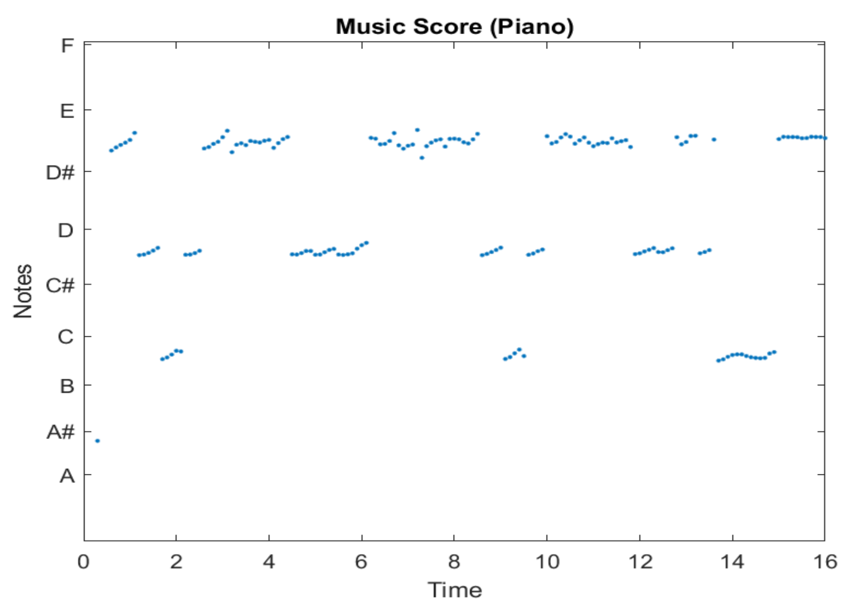

Figure 5: Music Score (Piano)

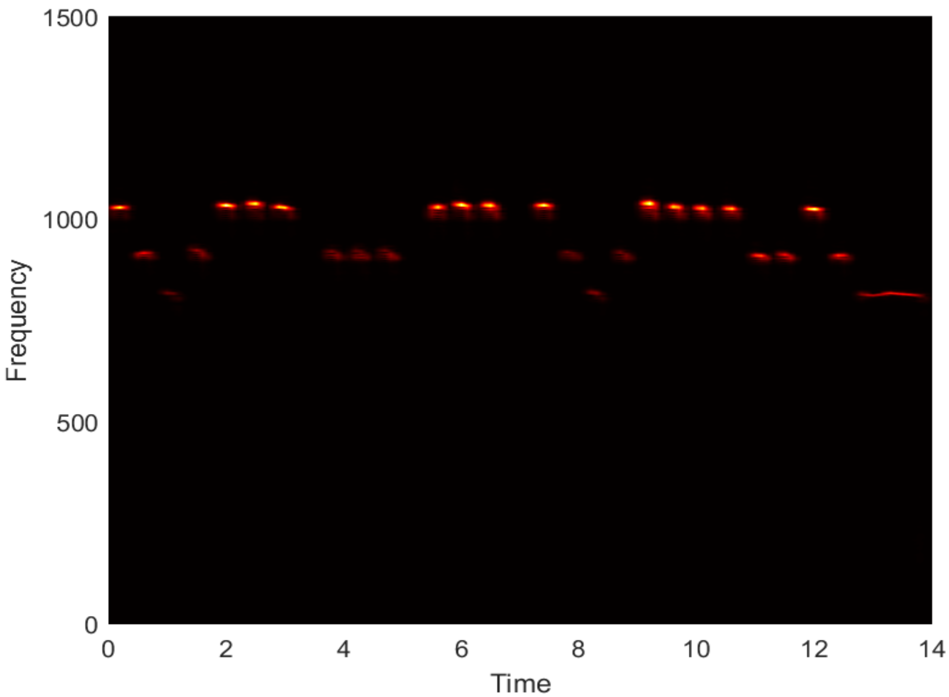

Figure 6: recorder spectrogram

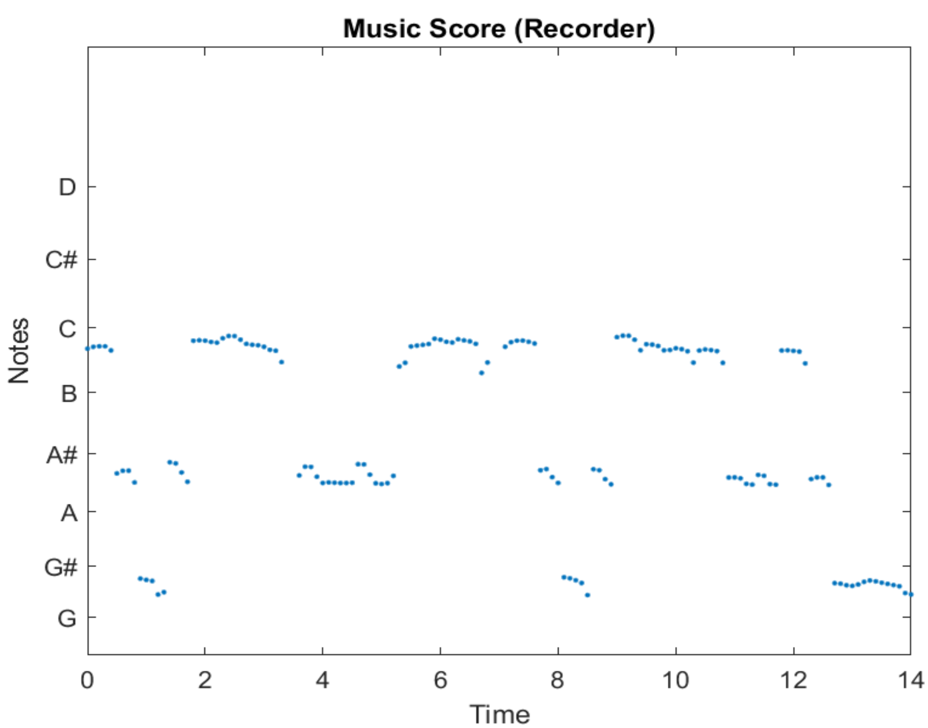

Figure 7: Music Score (Recorder)

Summary and Conclusions

We performed time frequency analysis on music files and looked at their spectrograms. To simultaneously obtain information in both the time and frequency domain, we used the gabor transform and looked at the effects of different parameters on the spectrograms. Finally, we obtained music scores corresponding to a song recorded on two different instruments and made various observations on the spectrograms.

Appendix A

MATLAB functions glossary (official documentation):

Y = fft(X) : returns the Fourier transform using a fast Fourier transform algorithm.

Y = fftshift(X) : rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array. In our work, we have used this for transforming data before plotting it.

Y = linspace(x_1**,** x_2**, n)** : generates n points. The spacing between the points is $\frac{(x_2-x_1)}{(n-1)}$.

APPENDIX B

Matlab Code

```
% Author: Aayush Chhabra
% Time-Frequency Analysis

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

load handel
v = y'/2;

% Let's look at the song handel
fig = figure(1);
plot((1:length(v))/Fs,v); hold on;
xlabel('Time[sec]');
ylabel('Amplitude');
title('Signal of Interest v(n)');
print(fig, '-dpng', 'fig1')

% Now, let's set up some variables for
% further analysis.

t2 = (1:length(v))/Fs;
t = t2(1:end-1);
n = length(t2); L = t2(end);
k = (2*pi/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);
S = v(1:end-1);

% Now, let's look at the frequency content of this data.
fig = figure(2)
St = fft(S);
plot(ks, fftshift(abs(St)))
title("Frequency content of the song (Fourier Transform)")
xlabel('Fourier Modes');
print(fig, '-dpng', 'fig2')
% Gabor Transform - sliding window.
sampling_control = 0.1; % = 0.01(over), 1(under), 0.1(normal)
tslide = 0:sampling_control:9;
% Parameters for gabor window
a = 1;
b = 1; % higher b means thinner window and vice versa

index = 1;
bvec = [1 10 20 40];
for b = bvec
    Sgt_spec = []; % Data collection for spectrogram.
    for center = tslide
        gabor = a*exp(-b*(t-center).^2);
        Sg = gabor.*S;
        Sgt = fft(Sg);
        Sgt_spec = [Sgt_spec; abs(fftshift(Sgt))];
```

```matlab

        figure(3)
        subplot(3,1,1)
        plot(t, S,'b'); hold on;
        plot(t, gabor, 'k'); hold off;

        subplot(3,1,2)
        plot(t, Sg, 'r'); hold on;
        plot(t, gabor, 'k');hold off;

        subplot(3,1,3)
        plot(ks, abs(fftshift(Sgt)), 'b');
        pause(0.01);
    end
    % Let's now use the data we have collected to make a spectrogram.
    fig = figure(4);
    subplot(length(bvec)/2, 2, index);
    pcolor(tslide,ks,Sgt_spec.');
    shading interp;
    colormap(hot);
    title(strjoin(["b =",b]));
    xlabel('Time');
    ylabel('Frequency');
    index = index + 1;
end
print(fig, '-dpng', 'fig4')

%% Analysis: Part II
%% Piano
clear; close all; clc;

tr_piano=16; % record time in seconds
y=audioread('music1.wav');
Fs=length(y)/tr_piano;
plot((1:length(y))/Fs,y);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Mary had a little lamb (piano)');
%p8 = audioplayer(y,Fs); playblocking(p8);

% Let's set up some variables for analysis
n = length(y);
L = tr_piano;
t2 = linspace(0, L, n+1);
t=t2(1:n);
S = y';
St = fft(S);
k = (1/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);

% Let's look at the frequency content
figure;
plot(ks, abs(fftshift(St)))

```

```
% Gabor Transform - sliding window.
sampling_control = .1; % = 0.01(over), 1(under), 0.1(normal)
tslide = 0:sampling_control:L

% Parameters for gabor window
a = 1;
b = 100; % higher b means thinner window and vice versa
piano_notes=[]; % Data collection for piano notes.
Sgt_spec = []; % Data collection for spectrogram.
for center = tslide
    gabor = a*exp(-b*(t-center).^2);
    Sg = gabor.*S;
    Sgt = fft(Sg);
    [m, ind] = max(Sgt);
    piano_notes = [piano_notes; abs(k(ind))];
    Sgt_spec = [Sgt_spec; abs(fftshift(Sgt))];

    figure(3)
    subplot(3,1,1)
    plot(t, S,'b'); hold on;
    plot(t, gabor, 'k'); hold off;

    subplot(3,1,2)
    plot(t, Sg, 'r'); hold on;
    plot(t, gabor, 'k');hold off;

    subplot(3,1,3)
    plot(ks, abs(fftshift(Sgt)), 'b');
    pause(0.01);
end
% Let's now use the data we have collected to make a spectrogram.
fig = figure(4);
pcolor(tslide,ks,Sgt_spec.');
shading interp;
colormap(hot);
xlabel('Time');
ylabel('Frequency');
ylim([0 500]);
print(fig, '-dpng', 'piano_spect')
%%
% Let's generate the music score for the piano
fig = figure(5);
plot(tslide, piano_notes, '.');
yticks([220.00, 233.08, 246.94, 261.63, 277.18, 293.66, 311.13, 329.63,
    349.23]);
yticklabels({'A','A#','B','C','C#','D', 'D#', 'E', 'F'});
ylim([200 350]);
title("Music Score (Piano)");
xlabel("Time");
ylabel("Notes");
print(fig, '-dpng', 'piano_score');
%% Recorder
clear; close all; clc;

```

```
tr_rec=14;
% record time in seconds
y=audioread('music2.wav');
Fs=length(y)/tr_rec;
plot((1:length(y))/Fs,y);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Mary had a little lamb (recorder)');

% Let's set up some variables for analysis
n = length(y);
L = tr_rec;
t2 = linspace(0, L, n+1);
t=t2(1:n);
S = y';
St = fft(S);
k = (1/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);

% Gabor Transform - sliding window.
sampling_control = .1; % = 0.01(over), 1(under), 0.1(normal)
tslide = 0:sampling_control:L;
% Parameters for gabor window
a = 1;
b = 100; % higher b means thinner window and vice versa
recorder_notes = [];
Sgt_spec = []; % Data collection for spectrogram.
for center = tslide
    gabor = a*exp(-b*(t-center).^2);
    Sg = gabor.*S;
    Sgt = fft(Sg);
    Sgt_spec = [Sgt_spec; abs(fftshift(Sgt))];
    [m, ind] = max(Sgt);
    recorder_notes = [recorder_notes; abs(k(ind))];
    figure(3)
    subplot(3,1,1)
    plot(t, S,'b'); hold on;
    plot(t, gabor, 'k'); hold off;

    subplot(3,1,2)
    plot(t, Sg, 'r'); hold on;
    plot(t, gabor, 'k');hold off;

    subplot(3,1,3)
    plot(ks, abs(fftshift(Sgt)), 'b');
    pause(0.01);
end
% Let's now use the data we have collected to make a spectrogram.
fig = figure(6);
pcolor(tslide,ks,Sgt_spec.');
shading interp;
colormap(hot);
xlabel('Time');
ylabel('Frequency');
```

```
ylim([0 1500])
print(fig, '-dpng', 'recorder_spect')

%% Let's generate the music score for the recorder
fig = figure(7);
plot(tslide, recorder_notes, '.');
yticks([783.99, 830.61, 880.00, 932.33, 987.77, 1046.5, 1108.7, 1174.4]);
yticklabels({'G','G#','A','A#','B','C', 'C#', 'D'});
ylim([750, 1300]);
title("Music Score (Recorder)");
xlabel("Time");
ylabel("Notes");
print(fig, '-dpng', 'recorder_score');
```