AMATH 582 Homework 2

Leon Yan

February 7, 2020

Abstract

This report serves to illustrate how we do the Short-time Fourier Transform(STFT) to analyze the temporal signal series in terms of not only the frequency components but also in which period they exist. In the report, first, we explore the results generated by different type of Garbor filter which varies in window's width, translation step size. Then, in application side, we analyze a song's audio signal and reproduce its music score, what's more, by comparing the Short-time Fourier Transform of the same music generated by a recorder and the piano, it's shown that, overtones makes the timbre of the instruments different. For the full materials including code and homework description, please go to url: `https://github.com/unitraveleryy/Amath582HW`

1 Introduction and Overview

In this homework, task I asks us to do the Short-time Fourier Transform(STFT) with different types of window and tune the hyper-parameters, like the width of the window, the step translation, and etc, to test their performance. Task II, in an applicable way, wants us to apply this technique to an audio of a song to reproduce the music score. Based on the recordings we have of different instruments' with regarding to the same music, we can observe that the overtones make the timbre different.

1.1 Data Description

In task I, the temporal audio signal Handel song is embedded in MATLAB and we can directly import that with command `load handel`. It is a 8.9249-second song with sampling rate equaling 8192 hz. In task II, **music1.wav** is a 16-second piano version of music "Mary had a little lamb" with sampling rate 43840 hz, and the **music2.wav** is a 14-second recorder version of the same period with sampling rate 44837 hz.

2 Theoretical Background

2.1 Short-time Fourier Transform, Garbor Filter

As we learned from our textbook [1], Garbor modified the Fourier transform kernel into $g_{t,\omega}(\tau) = e^{i\omega\tau}g(\tau - t)$ by adding $g(\tau - t)$ in the aim of localizing both time and frequency. The Garbor Transform, also known as the *short-time Fourier transform(STFT)* is then defined as the following:

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

where the bar denotes the complex conjugation of the function. Thus the function $g(\tau - t)$ acts as a time filter for localizing the signal over a specific window of time. 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.

Specifically, for convenience we will consider g to be real and symmetric with $||g(t)|| = 1$ and $||g(\tau - t)|| = 1$ where $||\cdot||$ denotes the L_2 norm. Thus the definition of STFT, can be simplified as:

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

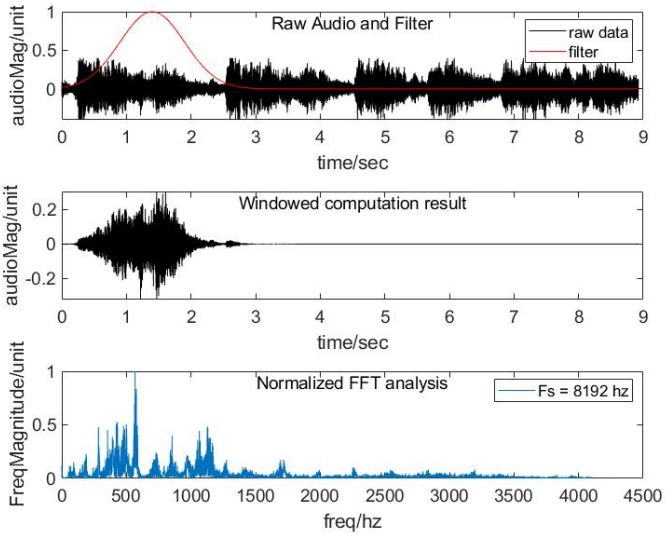

Figure 1: Demonstration of a generic window function composing on a temporal signal and the relevant analysis result.

In below we demonstrate several different types of local window:

1. Gaussian Filter: $g(\tau - t) = e^{-\text{wid}*(\tau-t)^2}$ specifies the local window at time t.
2. Mexican Hat Filter: $\phi(\tau - t) = (1 - (\frac{\tau-t}{\text{wid}})^2)e^{-\frac{(\tau-t)^2}{2\text{wid}^2}}$ specifies the local window at time t.
3. Shannon(step function) Filter:

$$g(\tau - t) = \begin{cases} 1, & |\tau - t| < 2 \cdot width \\ 0, & \text{otherwise} \end{cases}$$

An illustration of a generic Gaussian window can be seen in Fig. 1.

3 Algorithm Implementation and Development

Below is my algorithm implementation and development of the STFT:

1. Develop different type filter function
2. Specify the translation step
3. Apply and translate the filter with the signal to be processed
4. Store the FFT analysis of the filtered signal, construct the proper frequency array
5. Crop the spectrogram to just include the frequency of interests in different tasks
6. Plot the spectrogram

The relevant detailed implementation can be seen in Algorithm 1.

Algorithm 1: Denoising Multi-dimension Signal by FFT Algorithm

Input: sound data from `Handel, music1.wav, music2.wav`
Output: Spectrogram
Initialisation, data preparation :
Sgt_{spec} for storing the FFT results of the windowed signal
Determine the width, type and the translation step size of the window function
for $j = 1 : dt : t$ **do**
Construct the specified type window at the time translation equaling to j
Do the FFT analysis of the windowed time series
Crop and store the FFT analysis into the Sgt_{spec}
end for
Plot the spectrogram

4 Computational Results

Below are the computation results for each task.

Task 1. Using different types of window functions to get the STFT with varying the width of the window.

To ease the tension in storing the spectrogram data, in different tasks, we select the frequency range purposely to include the ranges we're concerned about. Specifically, in Task I, since men's and women's voice range from 138.59 $\sim$ 1046.5 hz, we just stored the frequency components between 130$\sim$ 1200 hz.

See in Fig. 2, for different window width of the Gaussian filter; in Fig 3, for different window width of the Mexican Hat filter; in Fig 4, for different window width of the Mexican Hat filter. The meaning of the width can be looked up at the Section 2.

See in Fig. 5, for the effects caused by different translation size with the Gaussian filter. The oversampling will improve the time temporal analysis of the frequency components, but the performance ceiling is bounded by the filter's width. Also, the drawback of the oversampling is that it will increasing the computation time. On the contrary, the downsampling will impair the time temporal analysis of the frequency components, i.e., it will dilate the spectrogram in time direction.

Task 2. Reproduce the music score of the song "Mary had a little lamb".

See in Fig. 6 for the demonstration of the STFT for the music temporal data in time series.

See in Fig. 7 for the spectrogram of the recorder and piano version of the song "Mary had a little lamb".

By utilizing the music score frequency distribution in the report, I deliberately crop the frequency range into 10 $\sim$ 5000 hz without removing the overtones. And pick the frequency corresponding the peak value based on the normalization windowed FFT analysis results.

What's more, the main notes in recorder differs from the piano version, which can be seen in Fig. 7. They have slight difference in keys.

Further, we can recognize that since the piano has more versatile overtones than recorder, the timbre of the piano is much more beautiful than the recorder.

5 Summary and Conclusions

As we can see in Fig. 2,Fig. 3 and Fig. 4, too wide window size will dilate the spectrogram in time direction, which is due to the fact that a wide window will be not able to capture the temporary frequency components; too narrow window size will dilate the spectrogram in frequency direction, which is caused by the fact that too narrow window will accommodate less sampling points, i.e., the frequency resolution is rough. This conclusion applies to all the filter types in this report.

As for different translation step size, too big translation step will dilate the spectrogram in time direction, because if the step size is greater than the width that window can cover, then this down-sampling action will lose the intermediate frequency information. Too small translation step size will improve the time temporal resolution of the spectrogram. The result of which can be seen in Fig. 5.

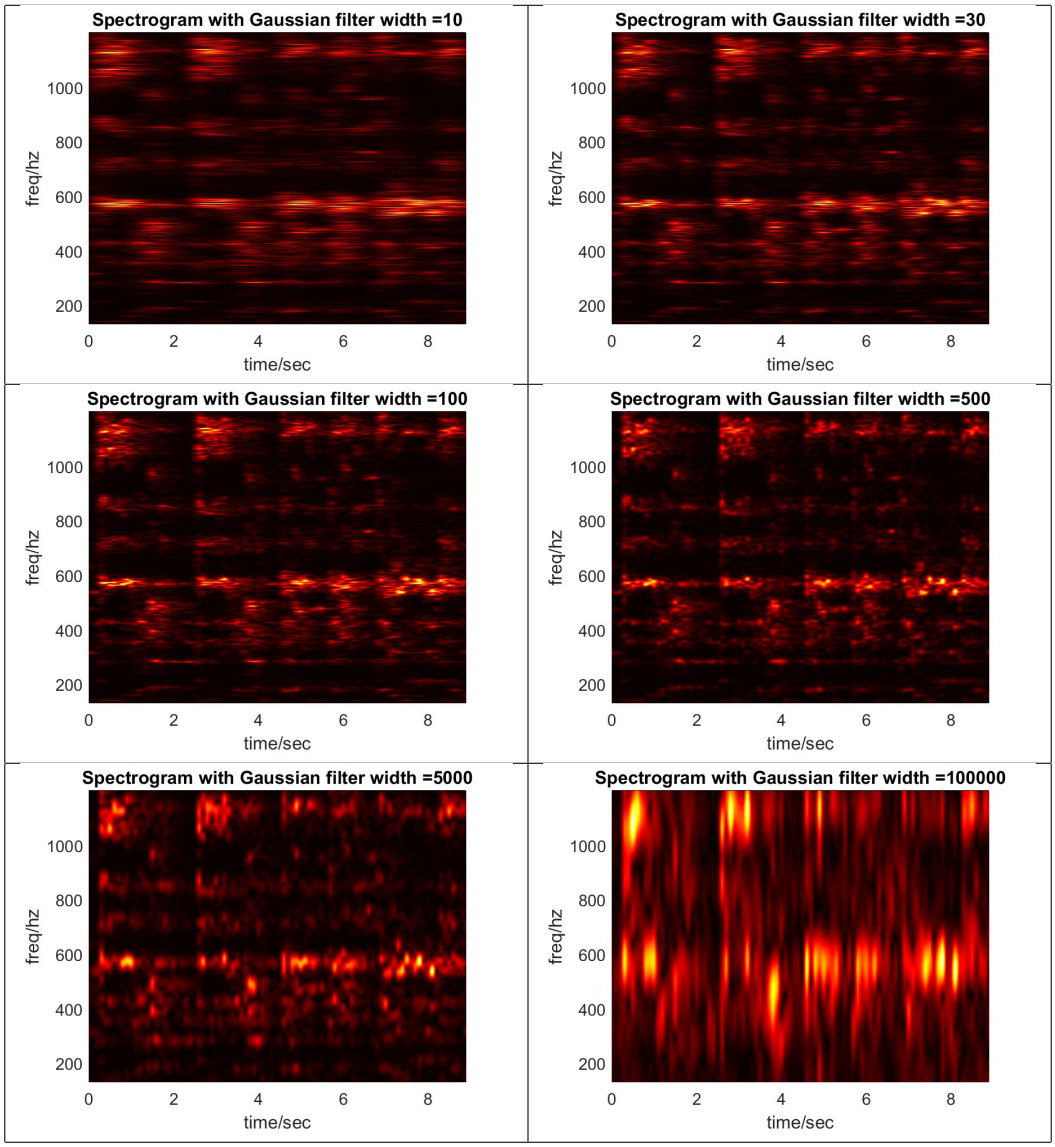

Figure 2: STFT with different Gaussian window width

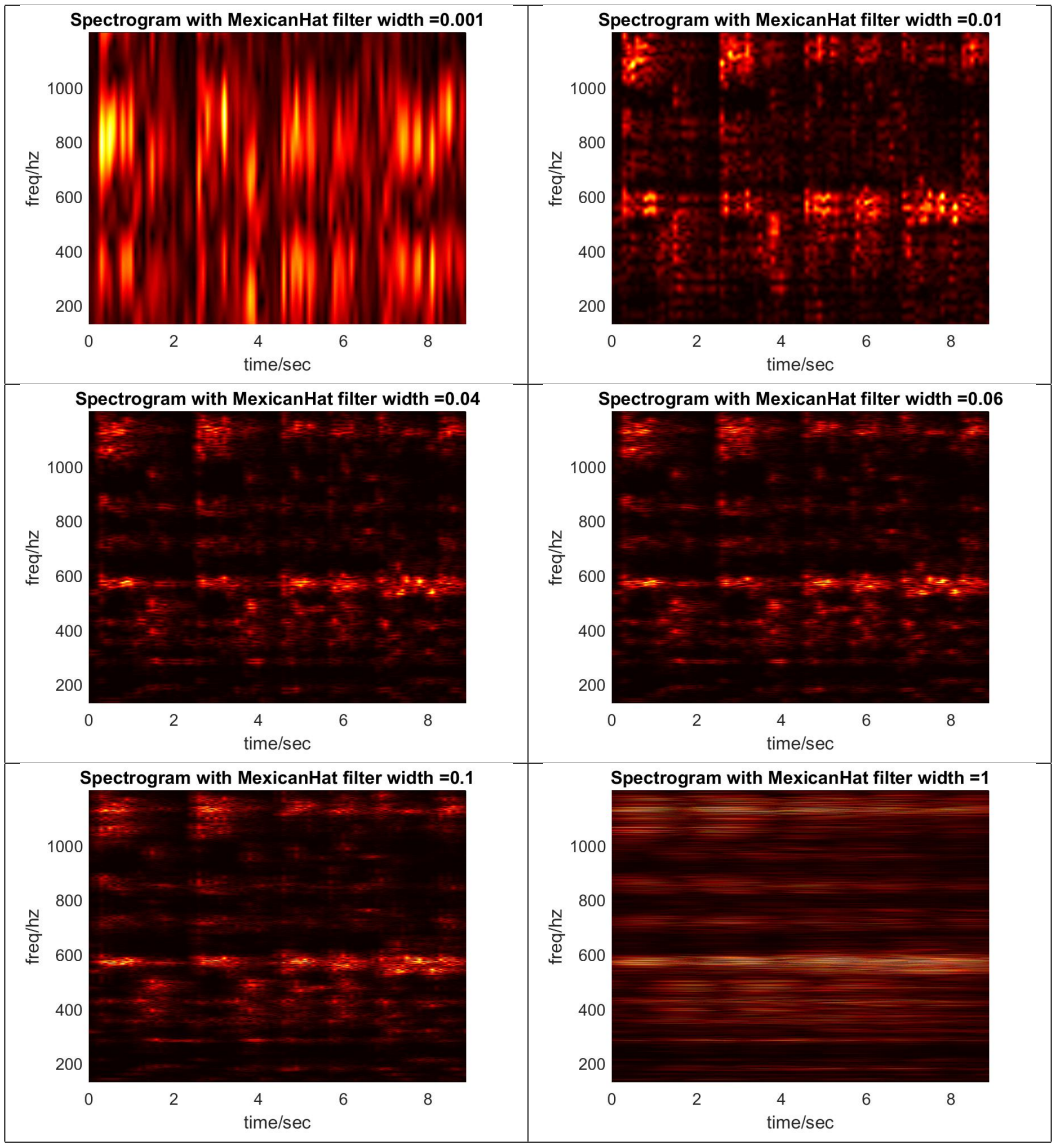

Figure 3: STFT with different MexicanHat window width

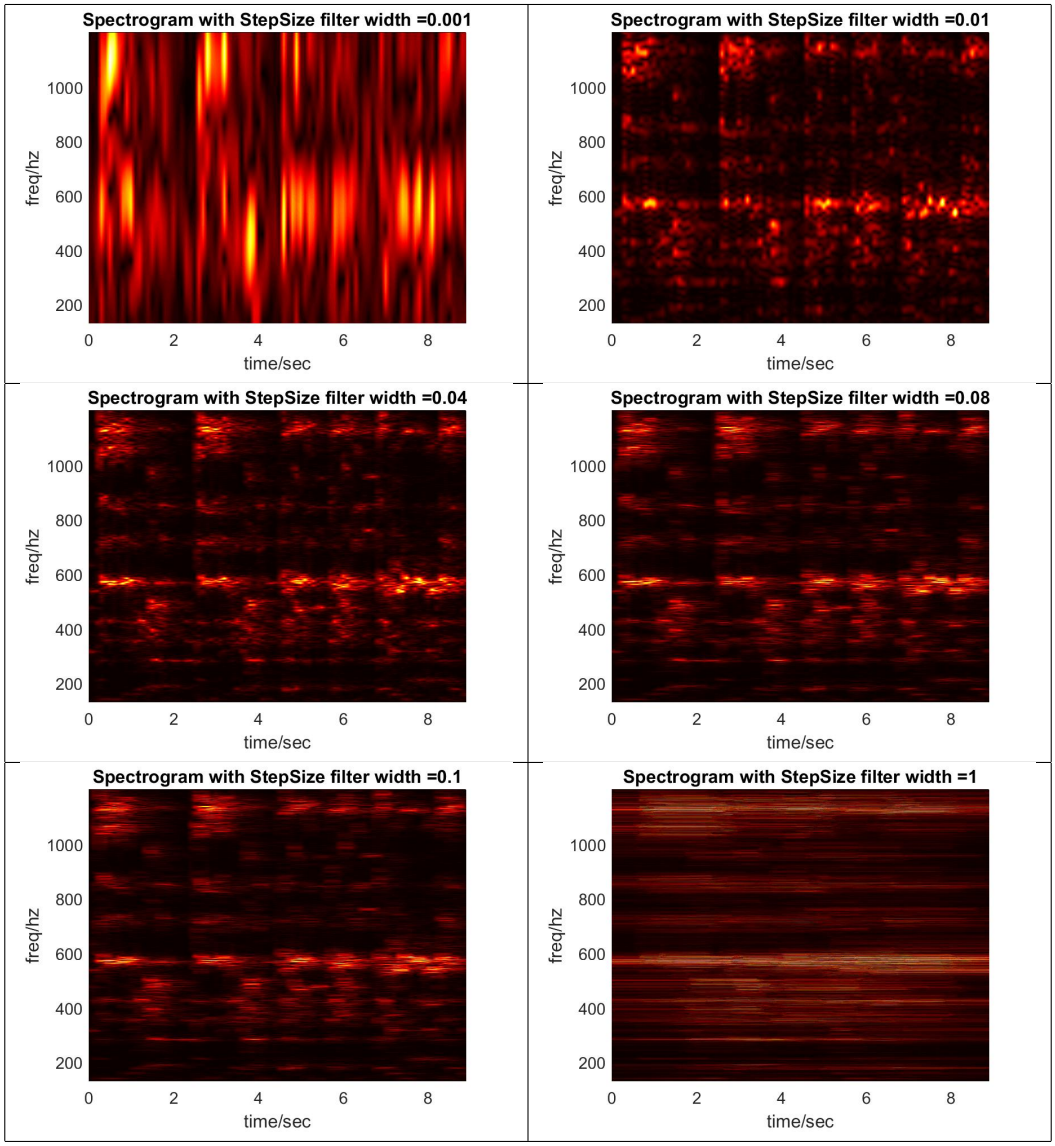

Figure 4: STFT with different StepSize window width

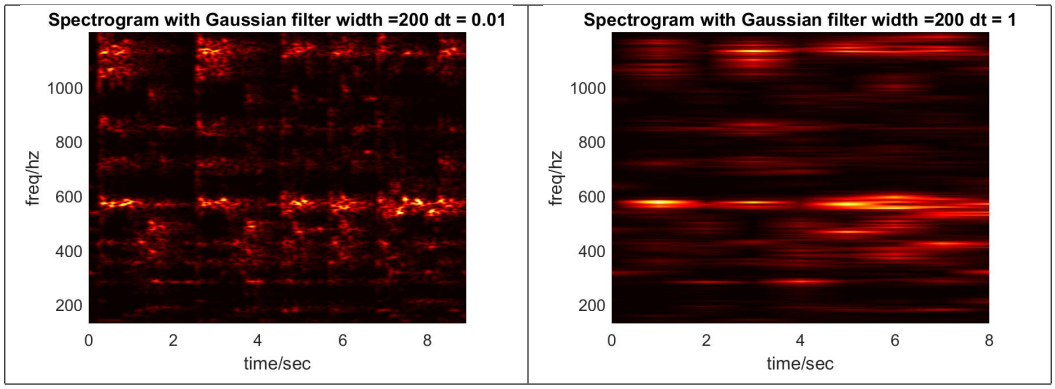

Figure 5: Too small translation step 0.01s (left); Too big translation step 1s (right)

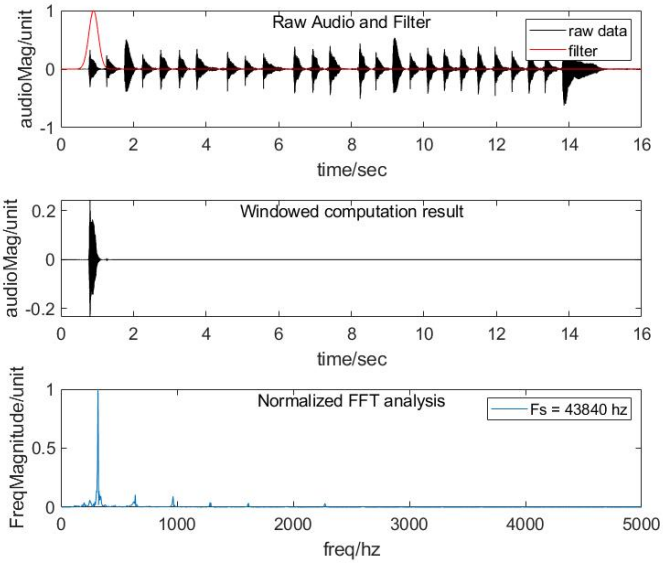

Figure 6: Demonstration of STFT on music data.

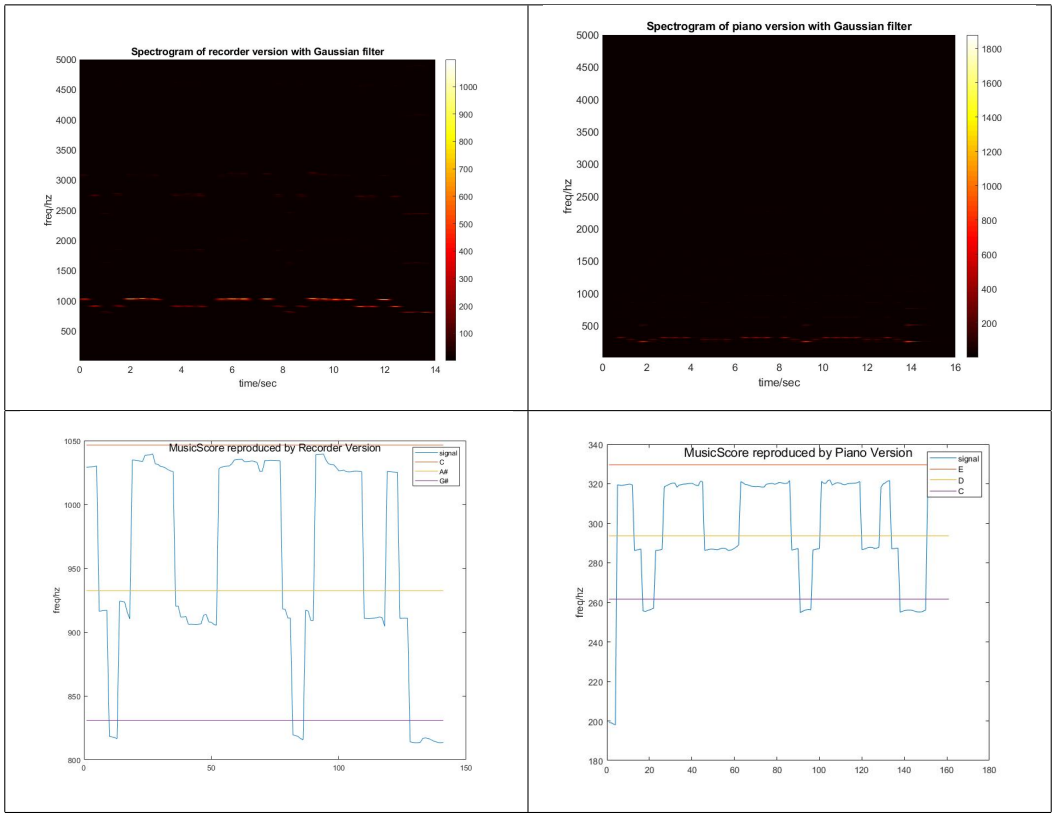

Figure 7: STFT of the piano and recorder version

The performance of different filter varies a little. To conclude, the Gaussian filter is the best among all of three, the StepSize window is the second, the reason why Mexican Hat performances not that well is due to the fact that it has some negative parts.

In task II, the Gaussian filter performs well to do the STFT for reproducing the music score of the song "Mary had a little lamb". And by comparing the STFT of the recorder version and the piano version, we can see that the more overtones the instrument has, the more beautiful timbre in the sound the instrument has.

Also, in Task II, the key in the piano version is semitone higher than the version of recorder.

References

[1] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

Appendix A MATLAB Functions

Below are the descriptions of the MATLAB function that I used in this report:

- `g=Windowfunction(width,tslide,t,"WindowType")` returns the filter type specified in argument "WindowType" with specified width.Current the function offered "Gaussian"(default), "MexicanHat" and "StepSize". `t` is the evenly sampled time vector of the signal to be processed. `tslide` is the translation length of the window with regarding to the start time of the signal.
- `Y = ceil(X)` rounds each element of X to the nearest integer greater than or equal to that element.
- `Y = fft(X)` computes the discrete Fourier transform (DFT) of X using a fast Fourier transform (FFT) algorithm.
- `ntitle(titlestring,varargin)` places a title within the plot instead of on top at the current graph. More example can be seen at `https://www.mathworks.com/matlabcentral/fileexchange/42114-ntitle`.
- `Y = abs(X)` returns the absolute value of each element in array X.
- `[x,y]=max(_)` finds the indices of the maximum values and returns them in array y, using any of the input arguments in the previous syntaxes. If the largest value occurs multiple times, the index of the first occurrence is returned.

Appendix B MATLAB Code

```
%% task 1
% try different width of the filter
% try different translation resolution of the Garbor filter
% try different types of filter
iptsetpref('ImshowBorder','tight');
load handel
v = y'/2; figure();
plot((1:length(v))/Fs,v);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');

% 0.01, 0.05, 0.1, 0.2, 0.5, 1
dt_array = [0.001, 0.01, 1, 0.05, 0.1, 0.2, 0.5, 1];
% dt = 0.01;
t = (1:length(v))/Fs;
S = v;
Nfft = length(t);
df = Fs/Nfft;
f = (0:(Nfft-1))*df;
f(f >= Fs/2) = f(f >= Fs/2) - Fs;
% frequency range in men and women voice
keep = (f >= 130 & f < 1200);

% width choices for Gaussian
% width = [0.2, 2, 10, 30, 100, 200, 500, 1000, 5000, 10000, 100000, 1000000];
width = [10, 30, 200, 500, 5000, 100000];
% width choices for MexicanHat
% width = [0.001, 0.005, 0.01, 0.04, 0.06, 0.1, 0.3, 1];
% width = [0.001, 0.01, 0.04, 0.06, 0.1, 1];
% width choices for StepSize Function
% width = [0.01, 0.04, 0.06, 0.08, 0.1, 1];
% width = [0.001, 0.01, 0.04, 0.08, 0.1, 1];
```

Listing 1: MATLAB code for Task1-seg1.

```
for ii = 2:3
dt = dt_array(ii);
tslide=0:dt:t(end);
for itr = 3:3
    wid = width(itr);
    Sgt_spec=[];
    for j=1:length(tslide)

        g = Windowfunction(wid,tslide(j),t,"Gaussian");
        Sg=g.*S;
        Sgt=fft(Sg);
        Sgt=Sgt(keep);
        Sgt= 2*Sgt;
        Sgt_spec=[Sgt_spec;
        abs(Sgt)];
        subplot(3,1,1), plot(t,S,'k',t,g,'r');
        xlabel('time/sec');ylabel('audioMag/unit');
        ntitle('Raw Audio and Filter','fontsize',14);
        legend('raw data','filter');set(gca,'FontSize',14);
        subplot(3,1,2), plot(t,Sg,'k')
        xlabel('time/sec');ylabel('audioMag/unit');
        ntitle('Windowed computation result','fontsize',14);
        set(gca,'FontSize',14);
        subplot(3,1,3), plot(f(keep),abs(Sgt)/max(abs(Sgt)))
        xlabel('freq/hz');ylabel('FreqMagnitude/unit');
        ntitle('Normalized FFT analysis','fontsize',14);
        legend('Fs = 8192 hz');set(gca,'FontSize',14);
        % axis([-50 50 0 1])
        drawnow
        pause(0.1)
    end
    figure()
    pcolor(tslide,f(keep),Sgt_spec.')
    xlabel('time/sec');ylabel('freq/hz');
    title(['Spectrogram with Gaussian filter width =', num2str(wid), ' dt = ' num2str(dt) ]);
    shading interp
    set(gca, 'FontSize', 14)
    colormap(hot)
end
end
```

Listing 2: MATLAB code for Task1-seg2.

```
%% task 2
tr_piano=16; % record time in seconds
y=audioread('music1.wav'); Fs=length(y)/tr_piano;
S = y';
t = (1:length(y))/Fs;
dt = 0.1;
Nfft = length(t);
df = Fs/Nfft;
f = (0:(Nfft-1))*df;
keep = (f >= 700 & f < 1200);
% keep = (f >= 10 & f < 5000);
% f(f >= Fs/2) = f(f >= Fs/2) - Fs;
tslide=0:dt:t(end);
width = [0.2, 2, 10, 30, 100, 200, 500, 1000, 5000, 10000, 100000, 1000000];
for itr = 4:4
    wid = width(itr);
    Sgt_spec=[];
    for j=1:length(tslide)
        g = Windowfunction(wid,tslide(j),t,"Gaussian");
        Sg=g.*S;
        Sgt=fft(Sg);
        Sgt=Sgt(keep);
        Sgt_spec=[Sgt_spec; abs(Sgt)];
        subplot(3,1,1), plot(t,S,'k',t,g,'r');
        xlabel('time/sec');ylabel('audioMag/unit');
        ntitle('Raw Audio and Filter','fontsize',14);
        legend('raw data','filter');set(gca,'FontSize',14);
        subplot(3,1,2), plot(t,Sg,'k')
        xlabel('time/sec');ylabel('audioMag/unit');
        ntitle('Windowed computation result','fontsize',14);
        set(gca,'FontSize',14);
        subplot(3,1,3), plot(f(keep),abs(Sgt)/max(abs(Sgt)))
        xlabel('freq/hz');ylabel('FreqMagnitude/unit');
        ntitle('Normalized FFT analysis','fontsize',14);
        legend('Fs = 43840 hz');set(gca,'FontSize',14);
        % axis([-50 50 0 1])
        drawnow
        pause(0.1)
    end
    figure()
    pcolor(tslide,f(keep),log(Sgt_spec.'))
    xlabel('time/sec');ylabel('freq/hz');title(['Spectrogram of piano version with Gaussian filter']);
    shading interp
    set(gca,'Ylim',[700 1200],'Fontsize',[14])
    set(gca, 'FontSize', 14)
    colormap(hot)
end
```

Listing 3: MATLAB code for Task2.