AMATH 582 Homework 2

Kyle Schultz*

February 7, 2020

Abstract

In this project, we explore the use of windowed Fourier transforms to gain insight into music. This work is in two parts: first we explore Gabor designs; second we apply the Gabor transform to recreate the music score of a given music piece. Parameters such as window width and sampling time were found to have significant impacts on the performance of the Gabor transform. We tweak these parameters and successfully apply them to recreate the music score from a song.

1 Introduction and Overview

1.1 Gabor Transform Design

In this part of the project, we explore design parameters of the Gabor transform and their impact their on the output. We apply these different Gabor transform designs on a 9 second sample of Handel's *Messiah*. The items we explore are:

1. Gabor window width
2. Translation time of Gabor window
3. Type of Gabor window (kernel function)

1.2 Music Score Reconstruction

After we explored how to design an effective Gabor transform, we apply this recreate the music of two pieces of music. Namely, we recreate the music score of *Mary Had a Little Lamb* played on both piano and recorder. This is achieved by first taking the Gabor transform, and then finding the frequency with the highest amplitude at each time. This gives us the primary frequency component over time (and thus the note played. However, we still need to find the time at which the note is played since the same note could be played multiple times. To find the time at which a note is played, we find the local minimum of the music amplitude. This is the time at which the signal transitions from decreasing to increasing and a note must be played to trigger an increase in music amplitude.

2 Theoretical Background

2.1 Gabor Transfrom

To classify the songs, we use the Gabor Transform which is a windowed Fourier Transform. The Gabor Transform gives us the spectral content at different time steps, which is the basis for how we compute the spectogram of our data. This is implemented using MatLab's `fft` command (appendix A). Figure 2 illustrates the general process.

*Code available on Github: https://github.com/kyleschultz0/

$$G_x(\tau,\omega) = \int_{-\infty}^{\infty} x(t) e^{-\pi(t-\tau)^2} e^{-j\omega t}\, dt \tag{1}$$

Note that in equation 1, a Gaussian is used to multiply the signal. The Gaussian acts as a window to localize the signal, so that we gain insight into the spectral content in specific time intervals. Note that other functions can other functions can be used as the Gabor window (see below).

2.2 Types of Gabor Window

Different filters can be used as the Gabor window depending on what properties are required. Three commonly used functions are Gaussians, Mexican hat wavelets, and Shannon filters (equations 1-3). These are also plotted in figure 1.

2.2.1 Gaussian

The Gaussian is often used as a filter because of its smoothness and good localization in time.

$$G(t) = e^{-\pi(t-\tau)^2} \tag{2}$$

2.2.2 Mexican Hat Wavelet

The Mexican hat wavelet is the normalized second derivative of the Gaussian. It has better localization properties than the Gaussian and is often used for edge detection.

$$\psi(t) = \frac{2}{\sqrt{3\sigma}\pi^{1/4}} \left(1 - \left(\frac{t-\tau}{\sigma}\right)^2\right) e^{-\frac{(t-\tau)^2}{2\sigma^2}} \tag{3}$$

2.2.3 Shannon Filter

The Shannon filter is also know as a step filter. It turns all signal components outside of the window to zero. It has great localization but is not smooth and can lead to discontinuities in the filtered signal. Its equation is below where σ is the window width.

$$\begin{cases} 1; & t - \sigma/2 \leq t \leq t + \sigma/2 \\ 0; & elsewhere \end{cases} \tag{4}$$

3 Algorithm Implementation and Development

3.1 Gabor Transform Design

In this section, our goal is to see how we can design effective Gabor transforms. The heart of this is the algorithm to compute to Gabor transform (algorithm 1)

Algorithm 1: Gabor transform

for $i = 1 : NumberWindows$ **do**
  Create filter centered at time i
  Apply filter to signal
  Apply `fft` to filtered signal
  Store windowed transform i
end for

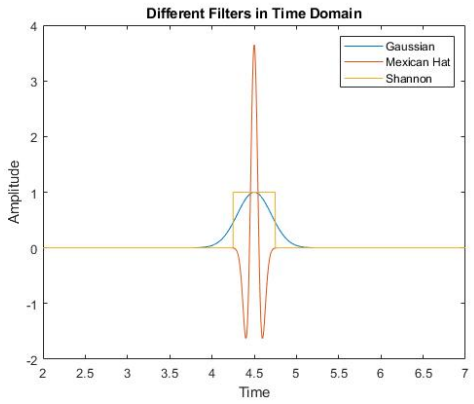

Figure 1: Different Gabor windows

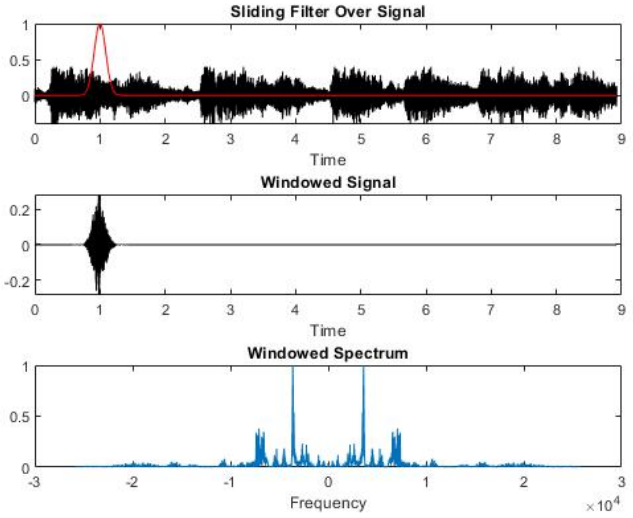

Figure 2: Process of computing Gabor transform

3.2 Music Score Reconstruction

1. Import songs using `audioread` (see appendix A)
2. Compute Gabor transform using algorithm 1
3. Find frequency of maximum amplitude in each Gabor window
4. Find local minimums of amplitude over time (time note is played)
5. Plot fundamental frequency and time notes played

4 Computational Results

4.1 Gabor Transform Design

The first item we explore is varying the window width of a Gaussian filter used as the Gabor kernel. The window width is represented by the parameter π in equation 1 (note that a larger π represents a finer window). As our window gets finer, we get better localization of the signal in time. As π increases in figure 3, there are fewer prominent frequencies at each time step (there is more black space in the spectograms). However, we lose low frequency content as low frequency signals will not fit in our window; note a dead band of low frequencies develops as π increases in figure 3. This loss of low frequency is not a concern in application since most musical instruments have a frequency above 50Hz, and therefore a window width of 0.02 seconds is sufficient to capture the frequency content. As we formulated the Gaussian in equation 1, the standard deviation will be:

$$\sigma = \sqrt{\frac{1}{2\pi}} \tag{5}$$

According to equation 5, with $\pi = 64$, $\sigma = 0.09 > 0.02$ and hence we will not lose any **relevant** low frequency content.

Next, we explore different types of filters: Gaussian, Mexican Hat, and Shannon. We normalize the filters to have the same window width by constructing the filters to have the same integral of their absolute value. The results are shown in figure 4, and are generally similar as might be expected given the window width is the same. That said, the Mexican hat wavelet has the best localization in time. We also see that

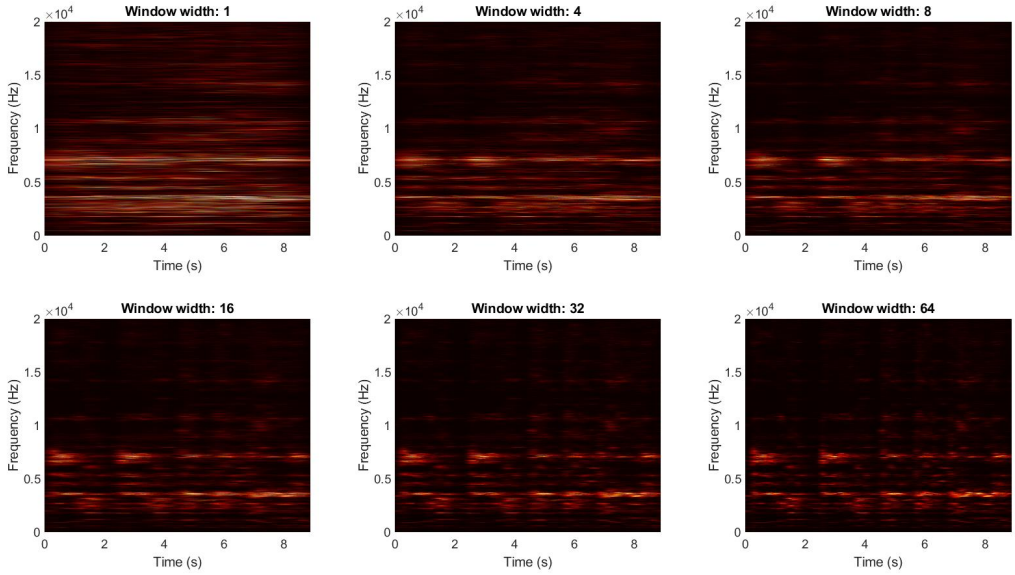

Figure 3: Varying width of Gaussian Gabor window

using a Shannon filter creates a 'jagged' spectrogram; there is more discretization of the frequencies in the spectogram. This is because the edges filter are not smooth like Gaussian or Mexican hat filters.

Lastly, we explore the effect of varying the translation time of the Gabor window in figure 5. We want a translation time small enough to capture music notes that are played. Music tempo is typically under 200 BPM, and hence even and eighth note in this fast tempo will be played for 0.15 seconds. Hence, we require a translation time less than approximately 150ms. If we decrease the translation time more, we gain nothing but increase our computational load. In figure 5, we see that the changing the sampling time from 10ms to 100ms causes no appreciable difference. If we increase our sample time too much (for instance 2000ms) we lose our localization in time. So, translation time should be chose to be sufficiently small depending on the tempo of the given song.

4.2 Music Score Reconstruction

We now use our Gabor filter developed in the first part of this project to recreate the music score for two pieces of music, *Mary Had a Little Lamb* performed on piano and recorder. Figures 6 and 8 show the spectrograms of these two songs. To recreate the music score, we must determine the principle frequency at each time instance. To achieve this, we find the frequency with the maximum amplitude over time. Next, we need to know when each note is played. We find out when each note is played by computing the local minimums of the frequency over time. The times when the notes are played are overlayed on the frequency at which they are played in figures 7 and 9. We now need to determine which notes the frequencies correspond to. Looking at the frequencies of musical instruments, we see that the high note is E, the middle note is D, and the low note is C. For the piano case, all the notes are about 10Hz lower than they should be; this discrepancy is likely due to the piano being slightly out of tune.

Also, note how there are faint signatures at higher frequencies for the piano. They occur at integer multiples of the fundamental frequency, and are known as overtones. The reason we had to find the maximum intensity signal at each time instance (instead of simply finding the average, for example) was that we needed to filter out these overtones to see the music score.

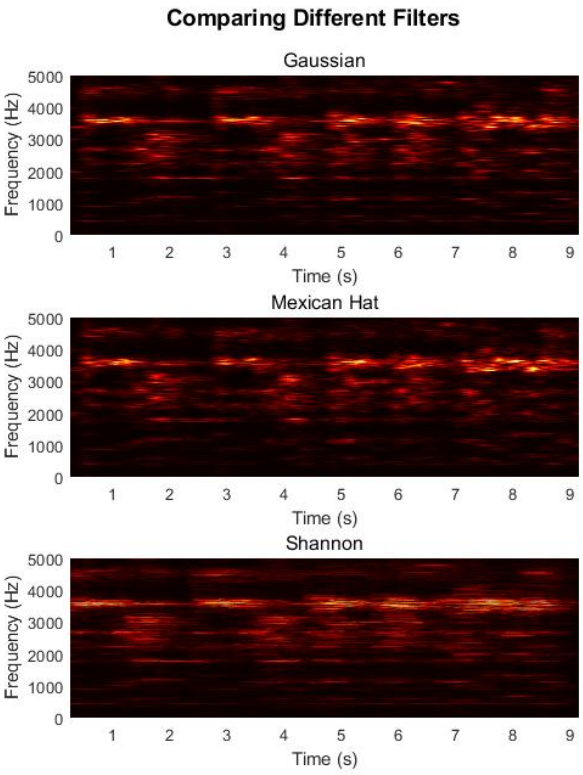

Figure 4: Testing different Gabor windows

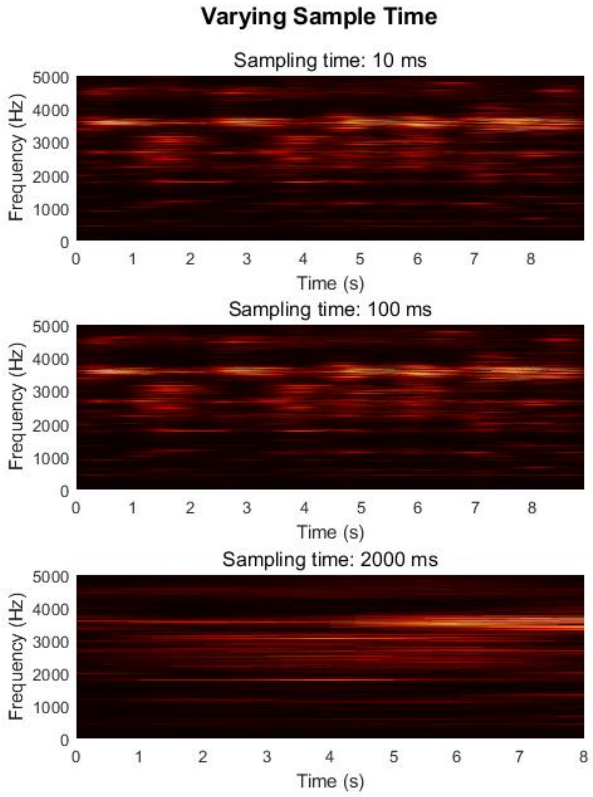

Figure 5: Varying translation time of Gabor window

5 Summary and Conclusions

In the first part of this project, we found some useful ideas for creating good Gabor transforms. We varied the window and showed that narrow windows have good localization in time but cause us to lose low frequency

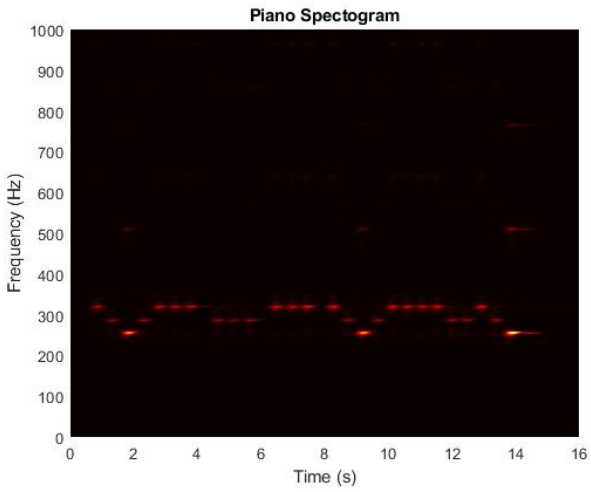

Figure 6: Spectogram of piano music

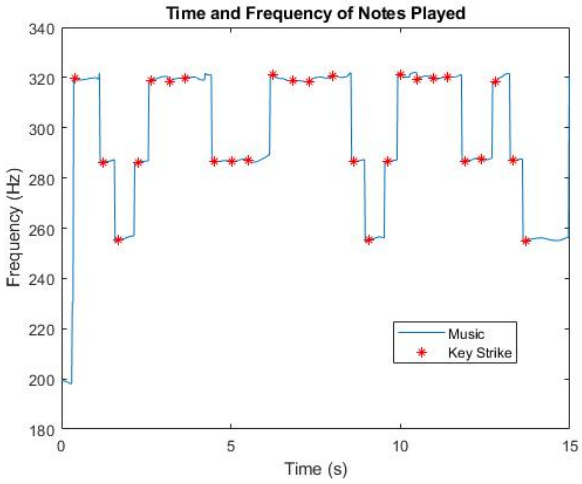

Figure 7: Music score. High note is E, middle note is D, low note is C.

]

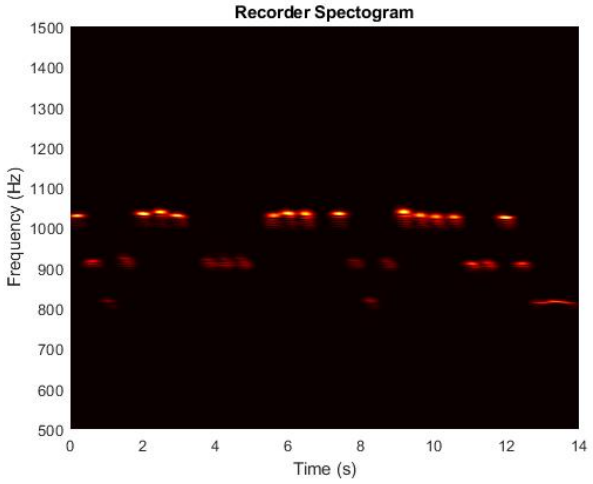

Figure 8: Spectogram of recorder music

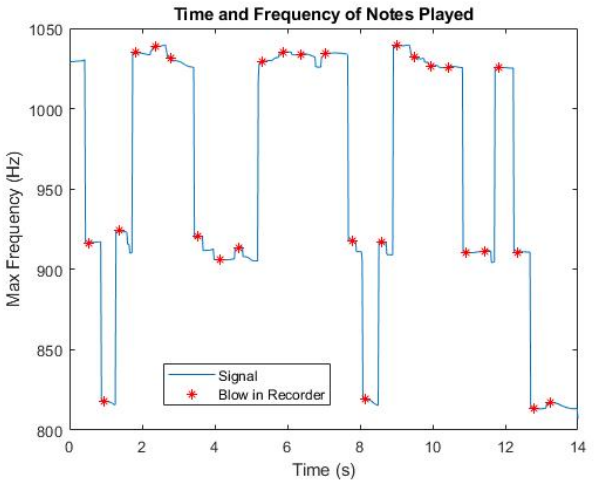

Figure 9: Music score. High note is E, middle note is D, low note is C.

content. However, music has a sufficiently high frequency that we did not have to worry about losing low frequencies. Next, we explored the translation time of our Gabor window. We found that oversampling had the effect of increasing computation time while having no impact on our spectogram. We also found that undersampling will cause poor localization in time. We also explored different Gabor kernels (filters) and found that the Mexican hat wavelet has better localization properties that a Gaussian.

Lastly, we used the Gabor filtering techniques to recreate a music score for two pieces of music. This illustrated the power of the Gabor filtering from the first part of the project.

Appendix A MATLAB Functions

Below are some of the most critical functions in our MATLAB implementation. Included are details pertinent to how we used them.

- `pcolor(tslide,ks,spectrum)` creates pseudocolor plot using the values in matrix `spectrum`, in our case the Gabor transform. It is defined by the x and y coordinates `tslide` and `ks`.
- `Un(:,:,:) = reshape(Undata(j,:),n,n,n` turns `Undata(j,:)` into a nxnxn array. Since `Undata` is 20x262144, we get a 64x64x64 matrix for each of the 20 time instances.
- $[y_i, x_i, z_i]$ `= ind2sub(sz, index)` converts a linear index, `index`, to the equivalent subscripts in an array of size `sz`. In our case, it converts the linear index into subscripts of a 64x64x64 array allowing us the extract the spacial location of the index.
- `Unt = fft(Un)` returns Fourier transform of the `Un` array using a fast Fourier transform algorithm.
- `Unt = fftshift(Unt)` rearranges a Fourier transform `Unt` by shifting the zero-frequency component to the center of the array. Namely, is shifts the transform so instead of going from 0 to L, it goes from -L/2 to L/2.

Appendix B MATLAB Code

```
clc; clear all; close all

%% Part 1
```

```
load handel
v = y'/2;
L=length(v)/Fs;
t2=linspace(0,L,length(v)+1); t=t2(1:length(v));
k=(2*pi/L)*[0:length(v)/2 -length(v)/2:-1];
ks=fftshift(k);
vt=fft(v);

p8 = audioplayer(v,Fs);
playblocking(p8)

tslide=0:0.1:L;
% tau = [1 8 64];
tau=64;
Sgt_spec_gauss=[];

Sp=spectrogram(v);
figure(1)
pcolor(abs(Sp))
shading interp
set(gca,'Fontsize',[14])
% set(gca,'Ylim',10^3*[0 20],'Fontsize',[14])
colormap(hot)

%% Effect of varying window width
for i=1:length(tau)
    for j=1:length(tslide)
        g=exp(-tau(i)*(t-tslide(j)).^2); % Gabor
        Sg=g.*v;
        Sgt=fft(Sg);
        Sgt_spec_gauss(:,j,i)=abs(fftshift(Sgt));
        subplot(3,1,1), plot(t,v,'k',t,g,'r'), title("Sliding Filter Over Signal"), xlabel("Time")
        subplot(3,1,2), plot(t,Sg,'k'), title("Windowed Signal"), xlabel("Time")
             subplot(3,1,3), plot(ks,abs(fftshift(Sgt))/max(abs(Sgt))), title("Windowed Spectrum"), xlab
        drawnow
        pause(1)
    end
end

figure(2)
for i=1:length(tau)
%     subplot(3,1,i)
    pcolor(tslide,ks,Sgt_spec_gauss(:,:,i)),
    shading interp
    set(gca,'Ylim',10^3*[0 5])
    title("Spectogram")
    colormap(hot)
    xlabel("Time (s)")
    ylabel("Frequency (Hz)")
end

%% Varying sampling frequency

t_sample = [0.01 0.1 2];
```

```
tau = 8;
Sgt_spec=[];
for i = 1:length(t_sample)
    tslide = 0:t_sample(i):L;
        for j=1:length(tslide)
            g=exp(-tau*(t-tslide(j)).^2); % Gabor
            Sg=g.*v;
            Sgt=fft(Sg);
            Sgt_spec(:,j,i)=abs(fftshift(Sgt));
        end
end

t_sample_label = [10 100 2000];

figure(3)
sgtitle("Varying Sample Time", 'FontSize' ,14,"FontWeight","bold")
for i=1:length(t_sample)
    tslide = 0:t_sample(i):L;
    subplot(3,1,i)
    pcolor(tslide,ks,Sgt_spec(:,1:length(tslide),i)),
    shading interp
    set(gca,'Ylim',10^3*[0 5],'Fontsize',10)
    title(sprintf('Sampling time: %d ms',t_sample_label(i)), "FontSize",12,"FontWeight","normal")
    colormap(hot)
    xlabel("Time (s)",'Fontsize',11)
    ylabel("Frequency (Hz)",'Fontsize',11)
end

tslide=0:0.1:L;

%% Mexican hat wavelet

% sigma = [4 1 0.5 0.25 0.1 0.05];
sigma=0.0565;
for i=1:length(sigma)
    for j=1:length(tslide)
        g_sobrero=2/(sqrt(3*sigma(i)).*pi.^(1/4)).*(1-((t-tslide(j))/sigma(i)).^2)...
            .*exp(-(t-tslide(j)).^2/(2.*sigma(i).^2)); % Mexican hat wavelet
        Sg=g_sobrero.*v;
        Sgt=fft(Sg);
        Sgt_spec_hat(:,j,i)=abs(fftshift(Sgt));
%         figure(4)
%         subplot(3,1,1), plot(t,v,'k',t,g_sobrero,'r')
%         subplot(3,1,2), plot(t,Sg,'k')
%              subplot(3,1,3), plot(ks,abs(fftshift(Sgt))/max(abs(Sgt)))
%         drawnow
    end
end

% for i=1:length(sigma)
%     figure(5)
%     subplot(3,3,i)
%     pcolor(tslide,ks,Sgt_spec_hat(:,:,i)),
%     shading interp
```

```
%     set(gca,'Ylim',10^3*[0 20],'Fontsize',[14])
%     title(sprintf('Window S.D.: %d s',sigma(i)))
%     colormap(hot)
% end

%% Shannon (step) filter

% t_width=[0.125 0.5 1];
t_width = 0.5;
width=t_width.*Fs;
for i=1:length(t_width)
    n=length(tslide);
    tslide = tslide+t_width(i)/2;
    for j=1:n
        g_shannon=zeros(1,length(v));
        g_shannon((tslide(j)*Fs-width(i)/2+1):1:(tslide(j)*Fs+width(i)/2+1))...
          =ones(1,width(i)+1);
        g_shannon=g_shannon(1:length(v));
        Sg=g_shannon.*v;
        Sgt=fft(Sg);
        Sgt_spec_shannon(:,j,i)=abs(fftshift(Sgt));
%         figure(6)
%         subplot(3,1,1), plot(t,v,'k',t,g_shannon,'r')
%         subplot(3,1,2), plot(t,Sg,'k')
%         subplot(3,1,3), plot(ks,abs(fftshift(Sgt))/max(abs(Sgt)))
%         drawnow
    end
end

% for i=1:length(t_width)
%     figure(7)
%     subplot(3,1,i)
%     pcolor(tslide,ks,Sgt_spec_shannon(:,:,i)),
%     shading interp
%     set(gca,'Ylim',10^3*[0 20],'Fontsize',[14])
%     title(sprintf('Window Width: %d s',t_width(i)))
%     colormap(hot)
% end

%% COMPARING SPECTROGRAM OF DIFFERENT FILTERS

figure(8)
sgtitle("Comparing Different Filters", "FontSize",14,"FontWeight","bold")
subplot(3,1,1)
pcolor(tslide,ks,Sgt_spec_gauss(:,:,1)),
shading interp
set(gca,'Ylim',10^3*[0 5],'Fontsize',[10])
title("Gaussian", "FontSize",12,"FontWeight","normal")
xlabel("Time (s)",'Fontsize',11)
ylabel("Frequency (Hz)",'Fontsize',11)
subplot(3,1,2)
pcolor(tslide,ks,Sgt_spec_hat(:,:,1)),
shading interp
set(gca,'Ylim',10^3*[0 5],'Fontsize',[10])
```

```
title("Mexican Hat", "FontSize",12,"FontWeight","normal")
xlabel("Time (s)",'Fontsize',11)
ylabel("Frequency (Hz)",'Fontsize',11)
subplot(3,1,3)
pcolor(tslide,ks,Sgt_spec_shannon(:,:,1)),
shading interp
set(gca,'Ylim',10^3*[0 5],'Fontsize',[10])
title("Shannon", "FontSize",12,"FontWeight","normal")
xlabel("Time (s)",'Fontsize',11)
ylabel("Frequency (Hz)",'Fontsize',11)
colormap(hot)

%% DEV ONLY

tau=12.56;
sigma=0.0565;
t_width=0.5;
width=t_width.*Fs;
tslide=4.5;

g=exp(-tau*(t-tslide).^2); % Gabor

g_sobrero=2/(sqrt(3*sigma).*pi.^(1/4)).*(1-((t-tslide)/sigma).^2)...
    .*exp(-(t-tslide).^2/(2.*sigma.^2)); % Mexican Hat

g_shannon=zeros(1,length(v));
g_shannon((tslide*Fs-width/2+1):1:(tslide*Fs+width/2+1))...
    =ones(1,width+1);
g_shannon=g_shannon(1:length(v));

figure(9)
plot(t,g,t,g_sobrero,t,g_shannon)
title("Different Filters in Time Domain")
legend("Gaussian", "Mexican Hat", "Shannon")
xlabel("Time")
ylabel("Amplitude")
xlim([2 7])




%% Reconstructing music score
%% Piano

tr_piano=16; % record time in seconds
y=audioread('music1.wav')'; Fs=length(y)/tr_piano;



t2=linspace(0,tr_piano,length(y)+1); t=t2(1:length(y));
```

```
k=(2*pi/tr_piano)*[0:length(y)/2-1 -length(y)/2:-1];
ks=fftshift(k);
tslide=0:0.1:tr_piano;
tau = 32;
max_freq=zeros(1,length(tslide));
amp = zeros(1,length(tslide));
Sgt_spec_piano = zeros(701440, length(tslide));



for j=1:length(tslide)
    g=exp(-tau(1)*(t-tslide(j)).^2); % Gabor
    Sg=g.*y;
    Sgt=fft(Sg);
    Sgt_spec_piano(:,j,1)=abs(fftshift(Sgt));
    [M,I]=max(abs(fftshift(Sgt)));
    max_freq(j)=-ks(I);
    amp(j) = M;
    %         figure(1)
    %         subplot(3,1,1), plot(t,y,'k',t,g,'r')
    %         subplot(3,1,2), plot(t,Sg,'k')
    %         subplot(3,1,3), plot(ks,Sgt_normal)
    %         xlim(10^3*[0 15])
    %         drawnow
end

amp_normal = amp/max(amp);
TF = islocalmin(amp_normal,'MinSeparation',0.2,'SamplePoints',tslide);
TF_shift = circshift(TF, 5);

E=329.63*ones(length(tslide));
D=293.66*ones(length(tslide));
C=261.63*ones(length(tslide));

figure(1);
plot(tslide, max_freq/(2*pi), tslide(TF_shift), max_freq(TF_shift)/(2*pi), 'r*');

legend("Music", "Key Strike")
xlabel("Time (s)")
ylabel("Frequency (Hz)")
title("Time and Frequency of Notes Played")
xlim([0 15])
hold off

figure(2)
plot(tslide, amp_normal, tslide(TF), amp_normal(TF), 'r*')
title("Points of Minimum Amplitude")
xlabel("Time (s)")
ylabel("Normalized Amplitude")
xlim([0 15])
%
% for i=1:length(tau)
%     figure(3)
%     pcolor(tslide,ks,Sgt_spec_piano(:,:,i)),
```

```
%     shading interp
%     set(gca,'Ylim',10^3*[0 20],'Fontsize',[14])
%     title(sprintf('Piano, window width: %d',tau(i)))
%     colormap(hot)
% end

figure(3)
pcolor(tslide,ks/(2*pi),Sgt_spec_piano),
shading interp
set(gca,'Ylim',10^3*[0 1])
title("Piano Spectogram")
colormap(hot)
xlabel("Time (s)")
ylabel("Frequency (Hz)")

%% Recorder

tau = 32;
tr_rec=14; % record time in seconds
y=audioread('music2.wav)'; Fs=length(y)/tr_rec;


t2=linspace(0,tr_rec,length(y)+1); t=t2(1:length(y));
k=(2*pi/tr_rec)*[0:length(y)/2-1 -length(y)/2:-1];
ks=fftshift(k);
vt=fft(y);
tslide=0:0.02:tr_rec;
max_freq=zeros(1,length(tslide));
amp = zeros(1,length(tslide));
Sgt_spec_rec = zeros(length(t), length(tslide));



for i=1:length(tau)
    for j=1:length(tslide)
        g=exp(-tau(i)*(t-tslide(j)).^2); % Gabor
        Sg=g.*y;
        Sgt=fft(Sg);
        Sgt_spec_rec(:,j,i)=abs(fftshift(Sgt));
        Sgt_normal=abs(fftshift(Sgt));
        [M,I]=max(Sgt_normal);
        max_freq(j)=-ks(I);
        amp(j) = M;
%         figure(3)
%         subplot(3,1,1), plot(t,y,'k',t,g,'r')
%         subplot(3,1,2), plot(t,Sg,'k')
%         subplot(3,1,3), plot(ks,abs(fftshift(Sgt))/max(abs(Sgt)))
%         drawnow
    end
end

amp_normal = amp/max(amp);
TF = islocalmin(amp_normal,'MinSeparation',0.2,'SamplePoints',tslide);
TF_shift = circshift(TF, 5);
```

```
figure(4)
plot(tslide, max_freq/(2*pi), tslide(TF_shift), max_freq(TF_shift)/(2*pi), 'r*');
legend('Signal', 'Blow in Recorder')
title("Time and Frequency of Notes Played")
xlabel("Time (s)")
ylabel("Max Frequency (Hz)")

figure(5)
plot(tslide, amp_normal, tslide(TF), amp_normal(TF), 'r*')
title("Points of Minimum Amplitude")
xlabel("Time (s)")
ylabel("Normalized Amplitude")

% figure(6)
% pcolor(tslide,ks/(2*pi),Sgt_spec_rec),
% shading interp
% set(gca,'Ylim',10^3*[.5 1.5])
% title("Recorder Spectogram")
% colormap(hot)
% xlabel("Time (s)")
% ylabel("Frequency (Hz)")

% xlim([0 15])

% for i=1:length(tau)
%     figure(4)
%     pcolor(tslide,ks,Sgt_spec_rec(:,:,i)),
%     shading interp
%     set(gca,'Ylim',10^3*[0 20],'Fontsize',[14])
%     title(sprintf('Recorder, window width: %d',tau(i)))
%     colormap(hot)
% end
```